Escola d'Enginyeria de Telecomunicació i Aeroespacial de Castelldefels
UNIVERSITAT POLITÈCNICA DE CATALUNYA

TREBALL FINAL DE GRAU

TÍTOL DEL TFG: Receptor Web-SDR a l'EETAC

TITULACIONS: Grau en Enginyeria de Sistemes de Telecomunicació, Grau en Enginyeria Telemàtica

AUTOR: Salvador Beltrán Obiol

DIRECTOR: David Garcia Vizcaíno

DATA: 1 de febrer del 2017

Títol: Receptor Web-SDR a l'EETAC

Autor: Salvador Beltrán Obiol

Director: David Garcia Vizcaíno

Data: 1 de febrer del 2017

Resum

En aquest projecte s'ha dissenyat, instal·lat i configurat un sistema de recepció de senyals d'RF utilitzant receptors SDR (Software Defined Radio) de baix cost que, mitjançant una interfície web, permet descodificar els senyals a qualsevol usuari del món -via internet- sense la necessitat de tenir un receptor de ràdio. Aquest sistema funciona a les bandes de radioaficionats i és multiusuari.

Aquest receptor online, ideat per a la seva utilització en les bandes de radioaficionats, també pot ser usat en qualsevol altra banda en funció de la seva configuració. Aquest projecte, és un servei més que proporciona l'escola de telecomunicacions, per tal de donar-se visibilitat a reu del món especialment entre els radioaficionats, els quals són els principals usuaris dels sistemes webSDR.
Les modulacions que suporta aquest sistema són:

- AM, FM, LSB, USB, CW

Per tal de rebre els senyals, s'han usat dos receptors SDR (Funcube Dongle Pro +, RTL-SDR) sintonitzats a les bandes de 144MHz (2m) VHF i 7MHz (40m) HF i s'han construït dues antenes per les dues bandes. L'antena d'VHF (2m) és una ground plane, i la segona antena per la banda de HF (40m) s'ha utilitzat una antena doble bazooka.

Tot el sistema estarà suportat per un PC, amb CPU Intel Pentium D a 3GHz i 2Gb de memòria RAM, que usarem de servidor. Aquest incorpora tot el programari necessari carregat en el sistema operatiu Ubuntu 14.04LTS allotjat sota la IP 147.83.115.237 i amb dos ports oberts per atendre les peticions (8072 i 8073) proporcionant accés a la informació adquirida pels diferents dongles.

Com a valor afegit a aquest projecte s'ha dissenyat un codi en Scilab per a desmodular senyals BPSK31, funcionalitat que avui en dia no es troba en cap programa webSDR.

Title: Web-SDR EETAC Receiver

Author: Salvador Beltrán Obiol

Director: David Garcia Vizcaíno

Date: 1 of February of 2017

Overview

In this project has designed, installed and configured a system of reception of signals of RF using SDR (Software Defined Radio) low cost receivers that by means of a web interface allow to decode the signals to any user of the world, via internet, without the need to have a receptor of radio, this system works to the amateur radio operator (ham) bands and is multiuser.

This online receptor contrived for the use in the ham bands but can be also be used in any one other band in function of his configuration, this project is another service that provides the school of telecommunications in order to give visibility around of the world especially between the hams than are the main users of the webSDR systems.
The modulations that bears this system sleep:

- AM, FM, LSB, USB, CW

To receive the signals have used two receptors SDR (Funcube Dongle Pro +, RTL-SDR)to the bands of 144MHz(2m) VHF and 7MHz (40m) HF and have built two antennas for the two bands. The antenna of VHF (2m) a ground plane and the second antenna for the band of HF (40m) has been a double bazooka antenna.

All the system will be supported by a PC, with a Intel Pentium D CPU at 3GHz and 2GB of RAM memory, that we will use as a server. This incorporates all the necessary software loaded in the operating system Ubuntu 14.04LTS lodged under the IP 147.83.115.237 and with two open ports (8072 and 8073) designated to provide access to the information acquired by the different dongles.

As added value for this project a BPSK31 signal decoder was designed on Scilab, this functionality is not present on any webSDR program yet.

Acknowledgment

I would like to thank Ariadna first for her patience and help.
Also to my friends Marta, David, Teresa, Mireia and Blanco for their support and their efforts to help me focus.
Thanks specially for the aid to Badr and Kushal for being with me under the rain in those moments of absolute stress.
Also to Xavi and Fran for his advice and encouragement. And all the EETAC technical area, specially to Juanjo and Raimon, for their advices and help.
And last but not least for my director for holding me and control himself in those moments that I know that he wanted to kill me.

Index

CHAPTER 0. INTRODUCTION & OBJECTIVES

In this project, we have designed a webSDR with 2 commercial dongles that is going to be hosted by EETAC. This receiver is used via web interface which means that any user around the world is going to be able to use it. For that reason, one of the needs of this project is that it must be a multiuser platform.

0.1 Introduction

In order to design the system, a previous overview of the actual SDR and ham market has been done, taking into account the most interesting characteristics of the actual SDR programs.

Once the market overview was done, the system properties and requirements can be determined. The last step is the system implementation.

To make this document clearer for the lector is divided on 4 chapters.

The first chapter is an introduction to the Software defined radio (SDR) in which the SDR market overview is included. The chapter will take a look to the SDR definition and the actual state of implementation in the Radio World.

On the second chapter, a more specific part of SDR will be exposed. This part is Web Software Defined Radio (WebSDR) that's the one in which the project is centered on. This part will also include its own market overview and specific introduction.

The third and most important part of the document, talks about the system design and implementation that will explain all the hardware and software, choosing stage and the further implementation.

The fourth and last chapter contains the results, conclusions and future lines of work. In this section all the results will be presented with the corresponding justification, other interesting results and lines of work.

The methodology used in order to develop the project followed a strict methodology that follow 3 different steps:

1- Define the requirements.
2- Implement in separate stages in order to check each one.
3- Implement and Integrate the final solution.

In order to build the system, some hams have been asked about requirements and other functionalities. This has helped to understand the most important factors, such as the most frequently used bands and modulations.

The project started with limited hardware, this fact limited the project capabilities. Just 2 dongles where available, this fact limits frequencies and sampling rate. Also the server is a refurbished EETAC PC which leads to the majority of errors that the development has encountered.

0.2 Objectives

The original project scope was to provide a webSDR receptor in order to make a system to help the hams to have access to the radiofrequency signals without the need of buying an expensive receptor.

This needs can be easily listed on the following points:

- Design a SDR web multiuser receptor operating the ham bands.
- Install a SDR web multiuser receptor operating the ham bands.
- Configure a SDR web multiuser receptor operating the ham bands.

In order to mount and design a low cost webSDR receiver we will have different commercial SDR receivers: FunCubeDonglePro+ and RTL-SDR (RTL2832U), antennas of HF/VHF and a PC.

Additionally we have designed a code in Scilab that allows the decoding of BPSK31, a digital modulation used by the hams, in order to integrate it with this system in a future project.

CHAPTER 1. SOFTWARE DEFINED RADIO (SDR)

In this chapter, the idea of SDR will be presented and an actual market overview will be shown, the idea is to take a look to the actual extension and use on the market of the SDR and the possibilities of this technology in the future.

1.1 Introduction to the software defined radio

Software Defined Radio (SDR) is a radio communication system which has a reconfigurable hardware part or component using specific software. For this propose, what makes one transceiver available to transmit / receive a lot of different frequencies just changing some parameters like the filter center frequency with the computer or adjust dynamically the gain of the amplifier, adjust the mixers and so on.

The number of SDR components on an actual transceiver is higher each day. This means that the costs are going to decrease following the actual dynamic because a single component can be used for many different functions, and this component will be used for more devices and vendors. This means that the capitalism system will ensure that this component can be easily accessible because as more extended is it's use, more vendors will enter in the game investigating about make the same function with a better design that costs less to be more competitive on the market, this will be extensible to all the SDR components. Also, the diversity of devices for the same function on different frequencies will be less important because of the most reconfigurable components, but will become more important on the precision of this devices because obviously if a device realizes a task on a higher range of frequencies is going to perform worst in a specific band than a specific component just for one band.

The happy idea of SDR is that a analogic to digital converter (ADC) for reception or a digital to analogic converter (DAC) for transmission can be attached to the antenna and all the signal will be processed with a Digital Signal Processor (DSP) that can be from a computer to a specifically designed DSP for some application as is shown in the fig. 3.1.

This scheme is not actually realizable due the problem that implies the ADC and DAC conversers for high baud rates.

The SDR was born thanks to the evolution of technologies in the final of 80's, some microprocessors were introduced on the transceivers in order to control some functions. Later, in the 90's, the DSP started to being developed what has allowed create digital filters and noise suppression. Was later, on the 90's, when Joseph Mitola started to create software defined radios, ~~that~~ was the first initiative to develop this devices, with the continuous technologic evolution, the radio equipment has been evolving from hardware radios to introduce each time more software defined components.

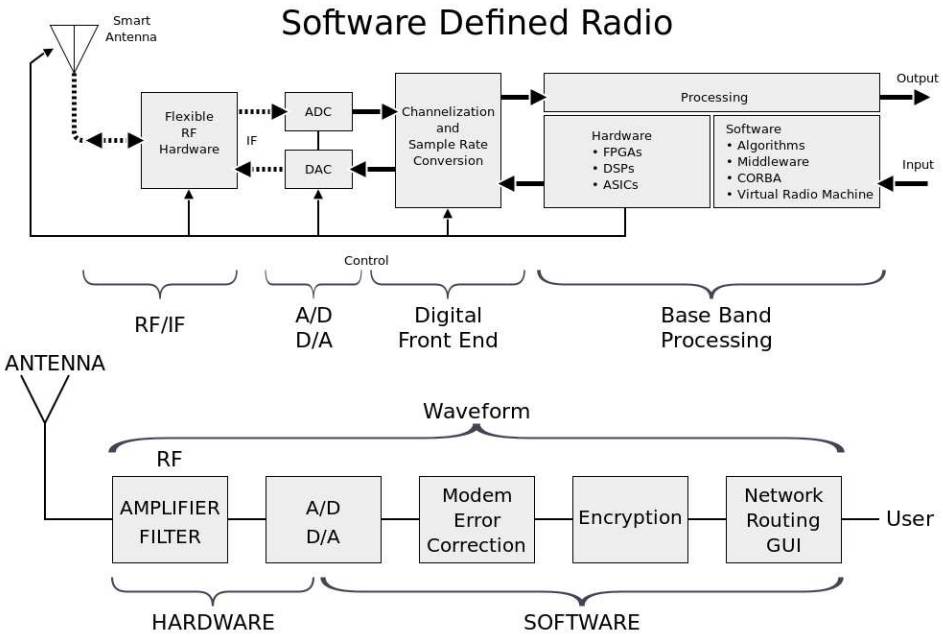


Fig 1.1 Ideal SDR scheme [2]

SDR Technology reached a boom from 1990 to 1995 with the SPEAKeasy military program, that wanted to build a radio for the United States of America air force that could operate from 2MHz to 2GHz, and between 2010-2012 for the amateur radio operators or hams because they realized that some of the usb digital video broadcasting - terrestrial (DVB-T) receivers that some business were selling have the capability of decoding the in phase and quadrature (I+Q) components of the signal. If they suppress the DVB-T demodulator blocks, the open source community discovered that the driver sends the digitalized baseband I+Q signal to the PC, where can be treated by software. Then, Osmocon decided to create the most popular library for control this kind of devices, also known as dongles, its name is librtlsdr [34].

The key point of this receivers is the price, which can be from 8 to 28 €, what makes every ham to be able to have its own receptor just developing his own code. Nowadays as we will see later, there is a high amount of free software in order to control this devices, what means that for just a few dollars you can have a receptor that some years ago can achieve a price higher than a 100USD.

1.2 Actual state of the software defined radio

Nowadays, almost all the ham radio equipment has an embedded DSP to process the signals, and are reducing the specific hardware components to the minimum expression that can be possible in order to maintain the same quality on the processed signal.

Almost, all the controls are digitals and make the digital processing of the signal after the ADC. Also, allows a higher reconfigurability of the receivers and transmitters, mainly the DSP can be found on the intermediate frequency (IF) and base band stages. Only the most expensive devices include a DSP for the RF final band because of the high speed that this DSP needs to achieve in order to process the signal at real time when working on high frequencies.

Other interesting functionalities that allows the DSP on the transmission (TX) and reception (RX) devices are the Automatic Gain Control (AGC) and the auto notch, or just implementing additional digital filters to have, in a more accurate way, the desired signal.

The DSP allows functions also not just in signal processing also in signal detection like the scan function, and the modulation detection [38] which are the most interesting and used functionalities, or just adjusting the squelch level.

This processors have also functionalities for the antenna tuning, some of the receivers and transmitters adjust automatically the output impedance depending on the transmitting or receiving frequency, what makes unnecessary to have a antenna adapter device between the antenna and the TX/RX device and adjust the antenna avoiding human failures and ensuring that the signal is transmitted with almost a perfect adaptation.

1.3 SDR Limitations:

SDR is limited by the quantization and sampling properties, that depend on the ADC or the DAC used on the receptor or transmitter.

SDR can improve the hardware radios by suppressing all the non-linearity's that the analog electronic components can introduce, but if the SDR program is bad implemented, as an example: the DSP algorithm, we can also introduce noise and harmonics to our digital signal. If we can convert parts or components of the receptors or transmitters to programs, we can achieve a better quality of reception/transmission by suppressing all the possible couplings and other hardware problems, we can get a more «clear» signal. But, on the other hand, a less specific hardware can have a worst «adaptation» to our signal. As less specific is the hardware, worst will be the quality of our analog signal, but once it became digital all the signals will be the «same», so we can treat them in the same way without taking care of having a special function, as it is need on the hardware for each signal depending on the frequency.

A key point is remember that most of the operations will be done in the IF stage, which allow us to have a limited range of incoming signals in our SDR receptor/transmitter.

Another important point of a SDR system on the reception part is the SNR, this will be modified on the ADC where we can simplify the maximum SNR formula if we take a sinusoidal waveform, as is shown on (1.1).

$$SNR_{MAX} = (1{,}76 + 6{,}02 \cdot N)dB \qquad \textbf{(1.1)}$$

Being N the number of bits of the ADC, so we can note that for each extra bit that we have on the ADC we can achieve until 6 dB more (1.2) which is the same as increase 4 times the power of the signal (1.3), so we are going to receive 4 times more power of our signal from the noise level.

$$SNR_{MAX} = (1{,}76 + 6{,}02 \cdot N + 1)dB = (1{,}76 + 6{,}02 \cdot N)dB + 6{,}02dB \quad \textbf{(1.2)}$$

$$6{,}02dB = 10^{(6{,}02/10)}W = 4W \quad \textbf{(1.3)}$$

This was a theoretical formula, but in the real live, we will always get a lower level. For that reason, we can say that we are not using all of the bits of the ADC, for that reason the effective number of bits become important. The effective number of bits (ENOB) is calculated always from the SNR, that has been measured before, so it's a practical quality parameter, not a theoretical one. For that reason, it's calculation is also based on practical measurements, the ENOB formula it's easily found isolating the N from the theoretical SNR formula (1.4).

$$SNR = (1{,}76 + 6{,}02 \cdot ENOB)dB \rightarrow ENOB = \frac{SNR[dB] - 1{,}76}{6{,}02} \qquad \textbf{(1.4)}$$

1.4 Analysis of some SDR software

In order to decode the raw I+Q samples that are sent from the dongles we need some software. That's why we will compare some of the more popular SDR software.

1.4.1 GLFER [18]

This application have some characteristics of spectrum analyzer for showing the acquired data from a RF receiver, and even has a transmission function which allows to send data using CW [19]. On this program, the most interesting functionalities are the automatic gain control on the spectrogram window and the high performance ARMA model, what makes the program interpret the input as noise plus sinusoids and tries to identify the sinusoids parameters like frequency, strength, and number of sinusoids [20].

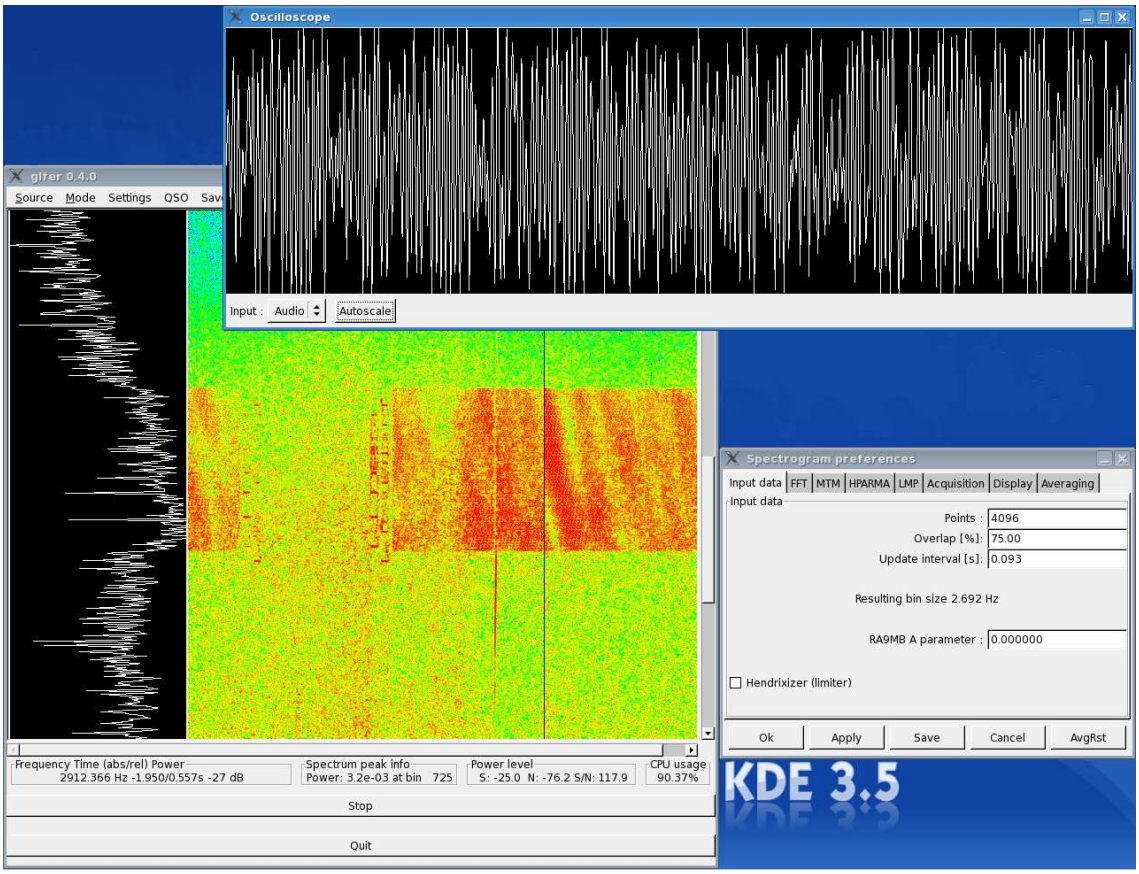


Fig1.2 GLFER screen capture with all windows opened.

1.4.2 SDR Sharp [52]

SDR Sharp is one of the most popular RTL-SDR controller software that allows the decoding of NFM, AM, LSB, USB, WFM, DSB, CW-L, and CW-U.

This software is really interesting because allows to the user to select the filter type, the center frequency, the audio gain or sample rate and other interesting functionalities like the auto tuner which is an useful tool similar to the normal scan function. In this case, the auto tuner search the most powerful signal that we are receiving and then, it tunes that signal in this options we have the peak level parameter, which is the received power of the signals from which our receiver must change.

Also, is interesting the automatic gain control (AGC) that allow us to maintain a signal more or less with the same strength, even if it suffers some attenuation and the reception level goes down. For that case, the AGC will increase the amplification level to compensate the attenuation.

Another interesting function of this software, is the spectrum or waterfall view and the option of combining both options, which is quite interesting because the spectrum analyzer can show in a more accurate way the received power. But, on the other hand, the waterfall display allows to have a temporal register of the received power on each frequency.

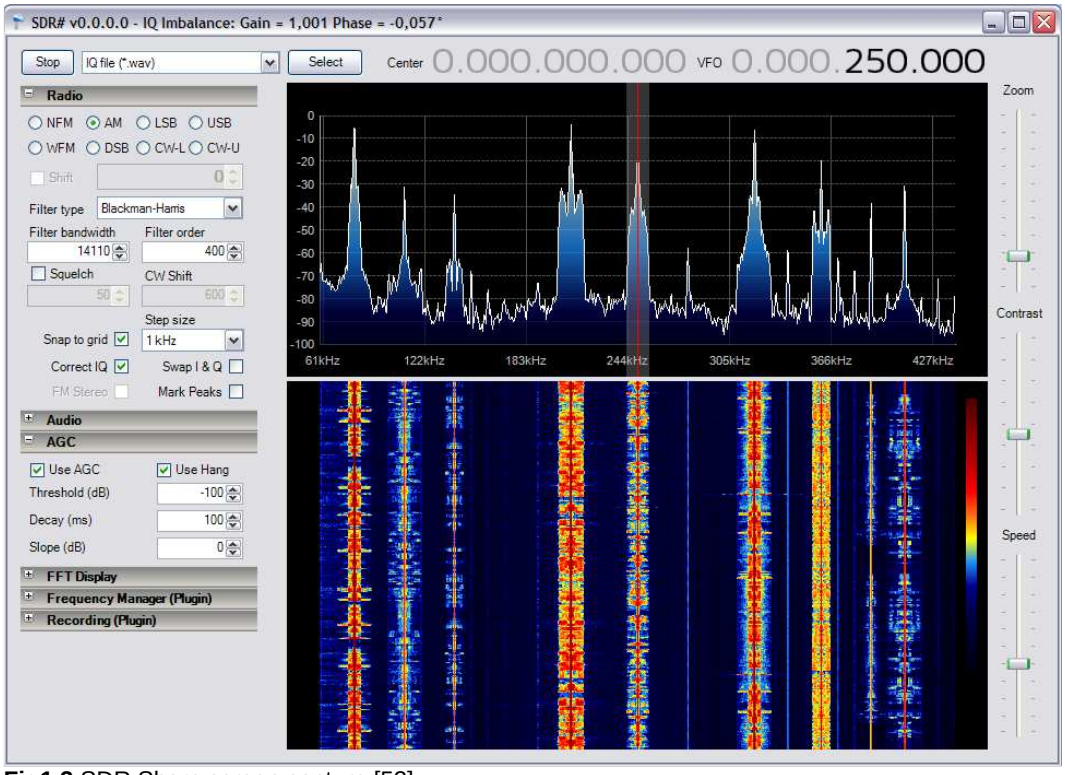


Fig1.3 SDR Sharp screen capture [52].

1.4.3 gMFSK [15]

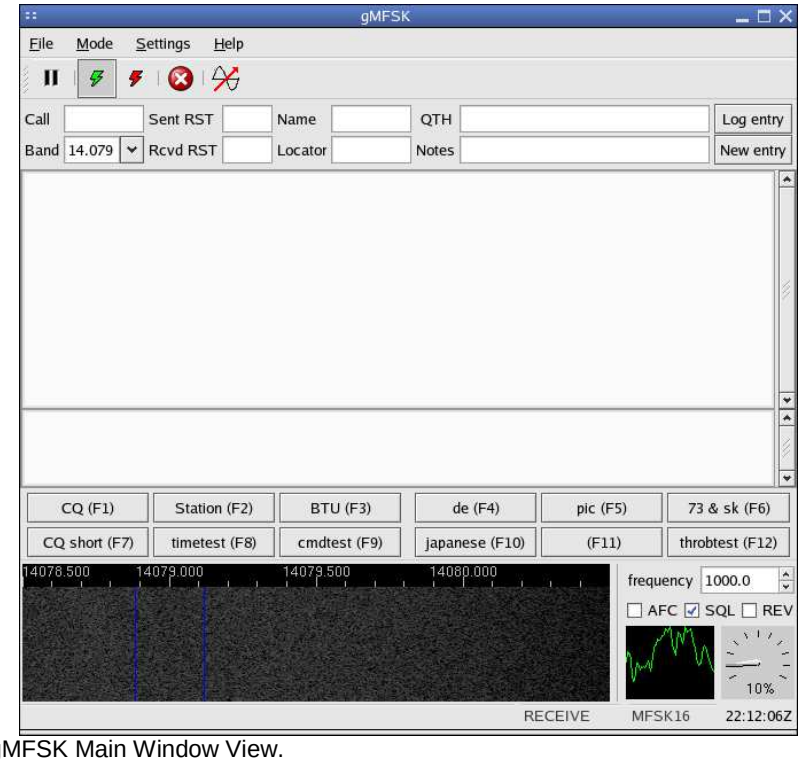


Fig 1.4 gMFSK Main Window View.

The gMFSK is a program for hams that allows the reception and transmission of lots of modulations, which are:

- MFSK (MFSK16 and MFSK8)
- RTTY
- THROB (1, 2 and 4 throbs/sec)
- PSK31 (BPSK and QPSK)
- PSK63
- MT63
- Feldhell

gMFSK uses the soundcard of the computer in order to become a SSB transceiver, using the main CPU for the signal processing and sending the final data as if it was a standard audio sample to the soundcard. This is an advantage because if the soundcard is supported by the main system, it will work which makes this program much more compatible with the equipment than if it needs to process data on the audio card.

As a detail, I would like to mention that gMFSK is free software licensed under the GNU General Public License, version 2. While I couldn't find the GLFER license, but I found the private software license of SDR Sharp.

1.4.4 FLDIGI [22][23]

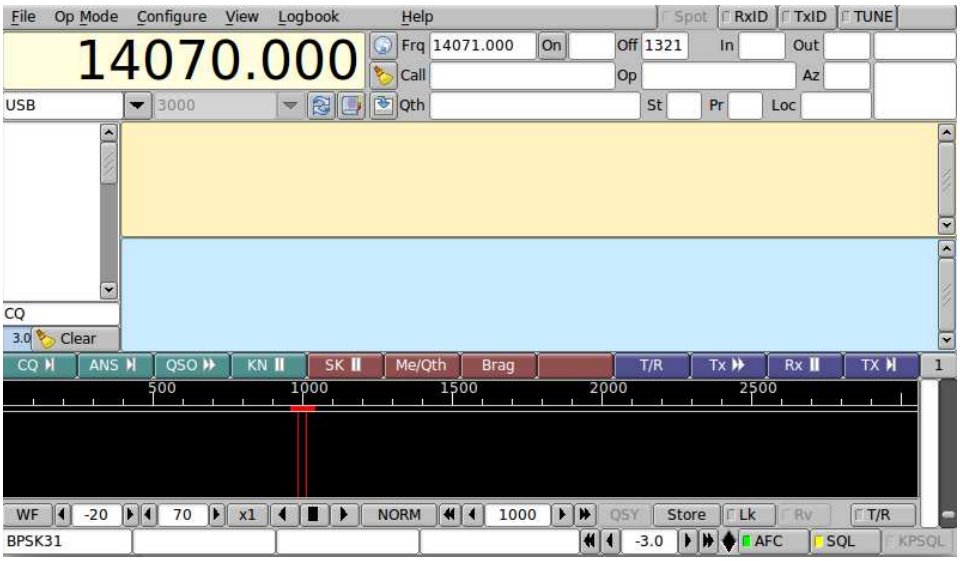


Fig 1.5 FLDIGI Main Window View.

FLDIGI is the most complete of all the programs that I have tested. It works as transceiver “intelligence”, it can control the receiver when it is directly connected to the radio allowing the transmission and reception of all kind of signals:

- Analog:
  - AM
  - FM
  - CW
  - WFM
  - CWR
  - USB
  - LSB
- Digital:
  - PSK
  - FSQ
  - IFKP
  - Contestia
  - DominoEX
  - Hellschreiber
  - MFSK
  - MT63
  - Navtex
  - Olivia
  - QPSK
  - 8PSK
  - PSKR
  - RTTY
  - SYNOP
  - THOR

- SITORB
- Throb / ThrobX
- WEFAX

This software is also released under the GNU Library General Public License version 2.

1.4.5 HAM RADIO DELUXE [24]

This program is developed for Windows, which means that, if you are running some other operative system (OS) that is not Windows, you are going to be unavailable to use it, except if you install some program like wine[25] for the GNU+LINUX distros that emulates a Windows OS on your GNU+Linux distro.

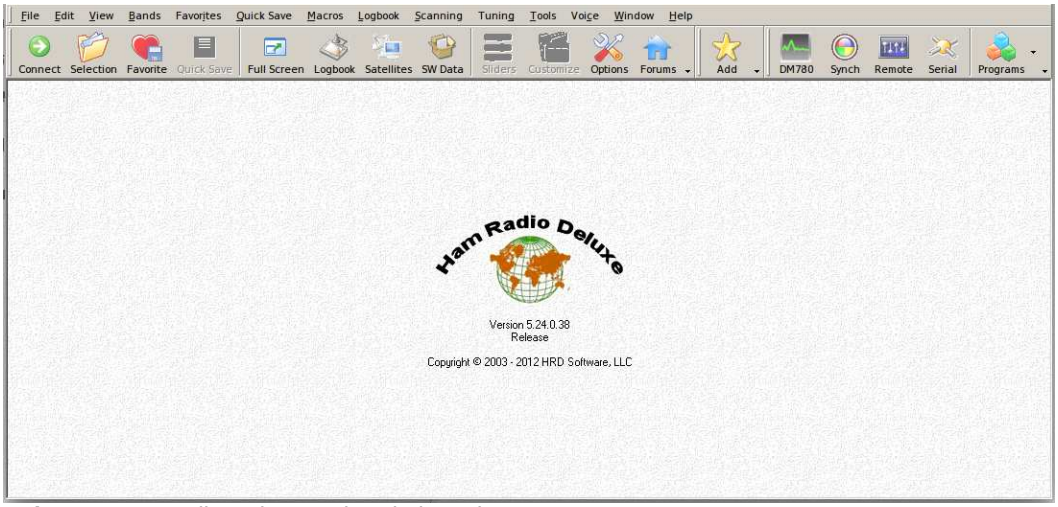


Fig 1.6 Ham Radio Deluxe Main Window View.

I want to say, that the version that we take a look of Ham Radio Deluxe is not the last one, is the last free (no economical cost) version that was published.

The modulations supported are the same as the FLDIGI ones.

Another interesting functionality, is the satellite tracking. It gives you the actual position of the satellite that you request. Also, allows you to direct automatically the antenna to the satellite in order to receive the better possible signal, positioning the antenna in the theoretically better orientation.

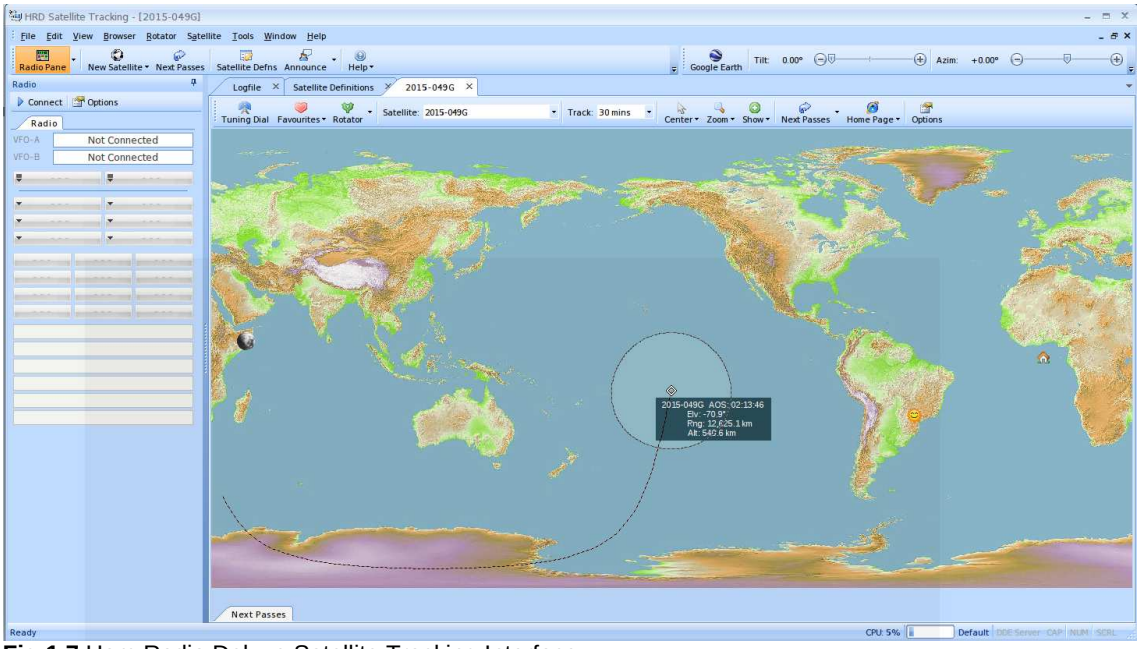


Fig 1.7 Ham Radio Deluxe Satellite Tracking Interface.

CHAPTER 2. WEB SOFTWARE DEFINED RADIO

Once the SDR concept is presented, a more specific SDR type will be explained: the Web Software Defined Radio (webSDR). This chapter is basic in order to take a fast look to the actual webSDR systems, which is the logical step before start our own design.

2.1 Introduction to Web SDR

This project has worked on an interesting possibility of the SDR, that is listening the RF signals without the need of having a receptor.

Not having a receptor does not mean not having a receiver, it means that is not necessary to have your own receiver. A receiver can be shared by multiple users, this is a quite interesting possibility knowing that it can have a main problem: if the receptor is tuned in one frequency and another user wants another frequency, this can be a difficult problem in the conventional receivers. Using SDR, we can process the signal with a DSP using the receiver to tune one band and send this entire band through a connection being each user of the receiver with his own DSP (usually using the PC as a DSP in order to filter the desired frequency inside the band and process it). This is obviously a happy idea as we will see later, but anyway that is the main idea.

2.2 Analysis of some WebSDR

In order to decide the characteristics of the EETAC WebSDR receiver, we have studied the actual webSDR market. Three different WebSDR software were found to analyze, taking a look also the programming language because it is one of the most important limitations that a WebSDR have in order to achieve real time.

2.2.1 Global Tuner [53]

This software was the “father” of webSDR and was released near 2000. It allows every registered user to upload a connection to his receiver and from the main webpage you can access to the desired receiver. This software is monouser because if you were in a receiver and you change the frequency, everyone that was connected to that receiver has also the frequency change.

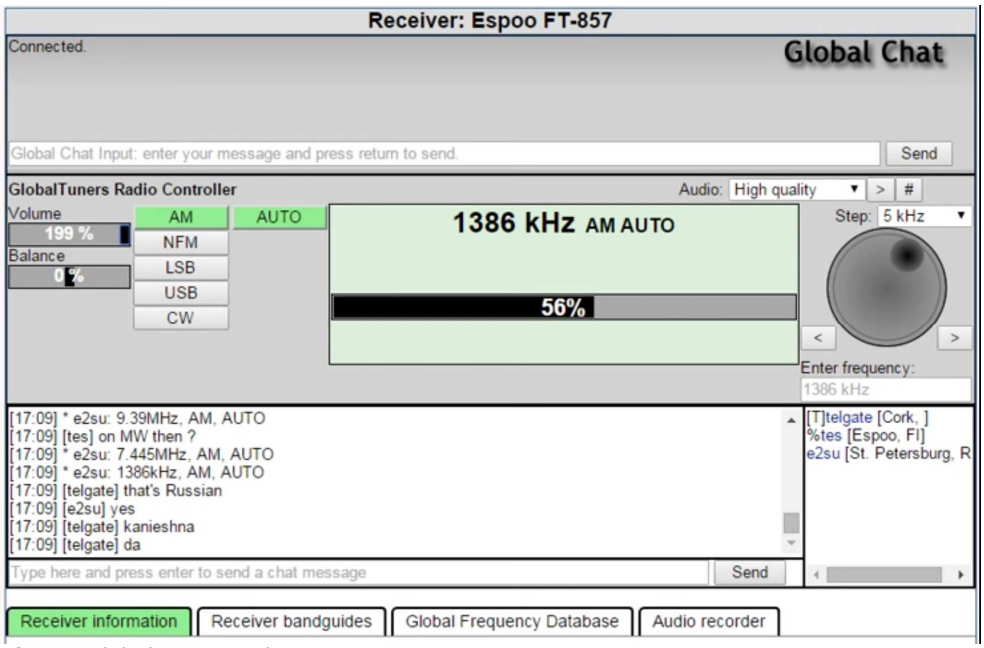


Fig 2.1 Global Tuner Web page.

2.2.2 WebSDR [26]

WebSDR is a Web application that takes the samples from the receiver in I+Q format and gives a waterfall to see all the spectrum. There you can put a filter in order to take the BW, that is useful for your signal and then you can choose the modulation desired.

An interesting functionality, is that you can add a label for some frequency, identifying the content what can be useful for some applications, like locating a desired public radio that transmit some interesting information or just tag the frequency in which you use to communicate.

This program is closed-source, that is why the server programming language can't be known, but the client part has a HTML interface that uses JS as «intelligence». Also, you can choose to make the page work with a java applet.

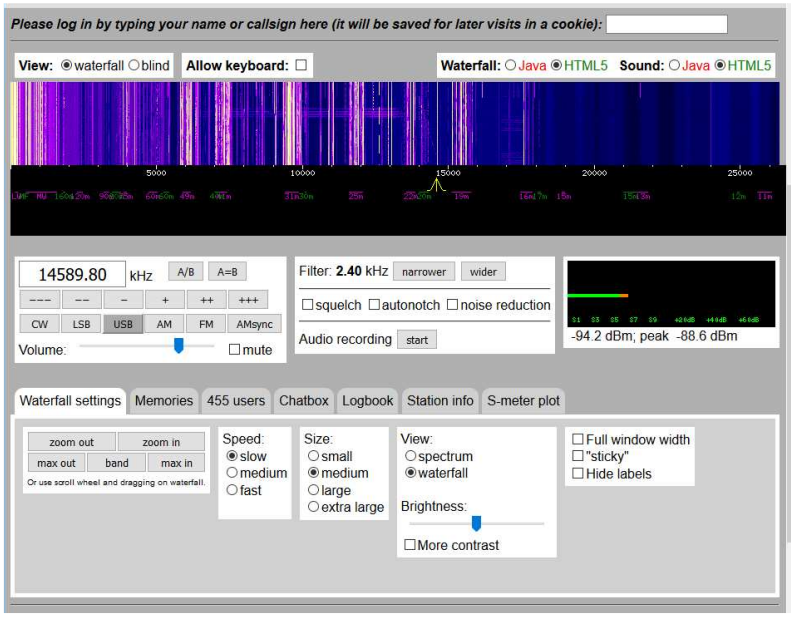


Fig 2.2 WebSDR Web page.

2.2.3 Shiny SDR [27]

This software is based on HTML5 client and Python server, it uses GNU Radio for processing the signals. It has some good features like allows multiple demodulators at once it's a bit unuseful as an open server. For that reason, is too difficult to find a server running shiny SDR as a public resource. This is because in order to connect it, requires a key, and once you are in it, gives you full control of the receiver. So, it's monouser, like Global Tuner.
Another disadvantage is that runs only over chrome as client, but by the other hand, all the GNU Radio gr-osmosdr compatible hardware can be used with this program.

Other interesting function is the logbook where you can save your favorite frequencies.

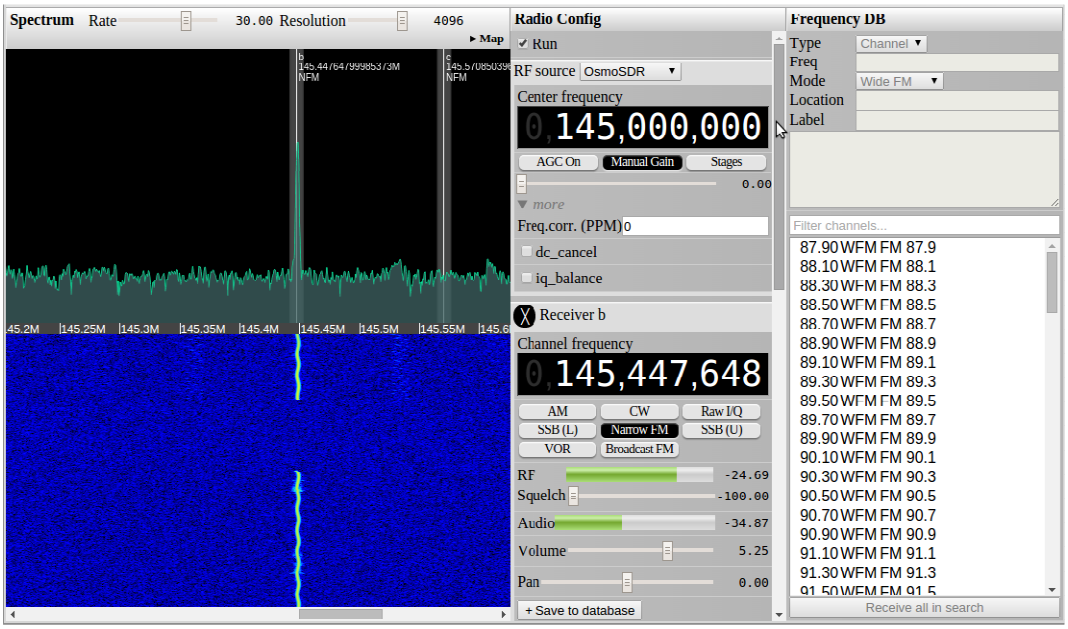


Fig 2.3 Shiny Web page.

2.2.4 OpenWebRX[37]

This WebSDR is implemented in python, which implements the server itself using some C code as DSP library and other functionalities. In the front end, it have html and JavaScript in order to run the waterfall and decode analogic modulations: AM, FM, LSB, USB, and CW.

It also allows change the squelch level dynamically or the amplification. Another good point is that this software, allows a visualization of server statistics as the number of clients, the server CPU and other client statistics as the audio buffer or the audio stream bit rate.
Also, comes up with a client log that shows up all the messages that you want to display to the user which can be quite useful on our system.

Fig 2.4 OpenWebRX Web page.

CHAPTER 3. EETAC WEBSDR RECEIVER

EETAC WebSDR was thought for ham users, which means that we will have to treat amateur radio signals that are mostly narrowband. However this is always depending on the used modulation.

3.1 EETAC WebSDR Hardware

The receiver in SDR must allow a wide frequency range, usually from KHz to MHz and contain a ADC that supports almost all of the signals that are transmitted by the RF devices. Normally, the ADC's included on the hardware work at 1Msps or more.

We are going to work with 2 different receivers that can treat different frequency ranges. For that reason, we will take a look to the possibilities of our receiver (we cannot change the receptors actually but we can take a look to the possibilities for a future improvement of the system).

We can compare easily all the receptors by putting all the main characteristics on a table that is the one that we can find below.

Table 3.1 Comparison of the most popular SDR receivers.

SDR	**Maximum RX sample rate**	**ADC resolution**	**Tuning range**	**Transmit capability**	**Price**
USRP B200 mini-i [28]	61,44 Msps	12 bits	70MHz-6GHz	Yes	745€
USRP N210[29]	25 Msps	14 bit	DC-6GHz	Yes	1810€
Nuand BladeRF X40 [30]	40 Msps	12 bits	300MHz-3,8GHz	Yes	400 €
Hack RF One [31]	20 Msps	8 bits	10MHz-6GHz	Yes	284 €
AirSpy [32]	10 Msps	16 bits	24Mhz-1750MHz	No	232 €
FunCube Dongle + [33]	192 Ksps	16 bit	150 Khz-240MHz 420 MHz - 1,9GHz	No	173 €
RTL-SDR [34]	2,4 Msps	8 bit	24MHz-220MHz*	No	12 €

* In the RTL-SDR it depends on the chip for the integrated circuit (IC) tuning should be completed with table 3.2

Table 3.2 Comparison of the tuning range for the most popular IC Tuners for RTL-SDR.

IC Tuner (Integrated Circuit Tuner)	Tuning Range
Elonics E4000	52-1100 + 1250-220MHz
R820T/R828D	24-1766MHz
FC0013	22-1100MHz
FC0012	22-948,6MHz
FC2580	146-308MHz+438-924 MHz

As we explained before, we are going to have 2 different receptors for 2 different bands, the first one is RTL-SDR with R820T and the second one is the Funcube Pro+.

In the following pages, we are going to analyze our receptors and their possible usages on our system.

3.1.1.1 RTL-SDR

Our RTL-SDR (Register-Transfer Level Software Defined Radio) has a maximum sample rate of 2,4Msps. Is the better one for the relation between quality/price, has a ADC resolution of 8 bits, and making some fast calculations, the maximum SNR that can achieve can be checked, almost 50 dB is the obtained value (2.5) what is quite good SNR for the money that we spent.

$$SNR_{MAX} = (1{,}76 + 6{,}02 \cdot 8)dB = 49{,}92dB \qquad \textbf{(3.1)}$$

The tuning range is marked by the IC tuner that has integrated, in our case, it allows a tuning range from 24MHz to 1766MHz. The main characteristics of the RTL-SDR are listed on table 3.1 and 3.2, but for simplicity, the most interesting characteristics are resumed on the 3.3 Table.

Table 3.3 Compilation of the used RTL-SDR main characteristics.

Maximum Sample rate	2,4 Msps
ADC Resolution	8 bit
Tuning Range	24-1766MHz

This RTL-SDR receivers are usually made with the idea of DVB-T demodulation, but the hams realized that is a way in which the DVB-T demodulator blocks can be turned off and therefore, they can get the raw I+Q data, as a example, when we use a software like librtlsdr, the DVB-T

demodulator blocks are automatically turned off and then we can obtain the raw signal. This practice has been highly extended since 2012, when the RTL-SDR project was released [34].

This dongle is composed by a RTL 2832 U and the R820T. The second IC was commented before so, is more interesting to explain the RTL2832U [35]. This is a high performance DVB-T coded orthogonal frequency-division multiplexing (COFDM) demodulator that supports a universal serial bus (USB). In its version 2.0, this IC digitalizes the baseband or the IF signal at 28,8Msps and contains a direct digital control (DDC) that uses a programmable FIR filter of 16 taps. However, the length of the filter, so the taps also, is limited by the lowest sample rate to be used without aliasing.

As was explained before, if librtlsdr is used, the built-in DVB-T demodulator is switched off and the I+Q samples are directly sent to the personal computer (PC) via bulk USB endpoint, which seems to have a limitation of 2,4Msps, if it sends with a higher sample rate the endpoint starts to drop samples. With this connections, we can approximate the RTL-SDR as shows the next schematic.

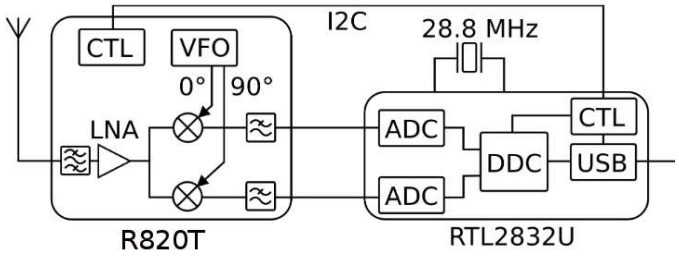


Fig 3.1 Used RTL-SDR schematic suppressing the DVB-T blocks [37]

This tuner will be one of the two used on the system and will be set on the 2 meters band which is a VHF band widely used by the hams.

3.1.1.2 Funcube Dongle Pro +

This is an interesting dongle, not only because of its characteristics, also by its story. This dongle was designed by some volunteers of the United Kingdom and Netherlands, which worked on a new amateur satellite, concept also known as the funcube project [44]. The dongle was born from the need of having a receptor for this nano satellite, and has become extremely popular between the hams.

Table 3.4 Compilation of the Funcube Pro + dongle main characteristics. [33]

Maximum Sample rate	192kHz
ADC Resolution	8 bit
Tuning Range	150kHz to 240MHz and 420MHz to 1.9GHz
TCXO (Temperature Compensated Crystal Oscillator)	0.5ppm (in practice about 1.5ppm)
Eleven discrete hardware front end filters	
Front end LNA OIP3	30dB

As we can see, this dongle has more interesting characteristics than the RTL-SDR one, beginning by the frequencies that can achieve, that obviously are more, it has a better frequency range and a better ADC resolution what means that we will get a better SNR, but on the other hand it's interesting to note that the sample rate is lower.

It works quite good in spite of all the critics that some people did on some web sites (you can find an example reading the comments of the characteristics in the dongle official page), the 16 bits ADC is quite good because of the most of computers use to confuse it as a sound card what makes easier to be compatible with the computer, and with more bits on the ADC a higher maximum SNR can be achieved as is shown in (3.2).

$$SNR_{MAX} = (1{,}76 + 6{,}02 \cdot 16)dB = 98{,}08dB \qquad \textbf{(3.2)}$$

Is also heard that this dongle provides a real bandwidth of 48 KHz instead of 192 KHz in the windows operative system (OS) due some errors of the controller software for this dongle.

This receptor should be much better than the RTL-SDR, in fact it gives much more interesting functionalities, and the main scheme for this receptor really simplified is something like the schematic shown on Fig 3.2.

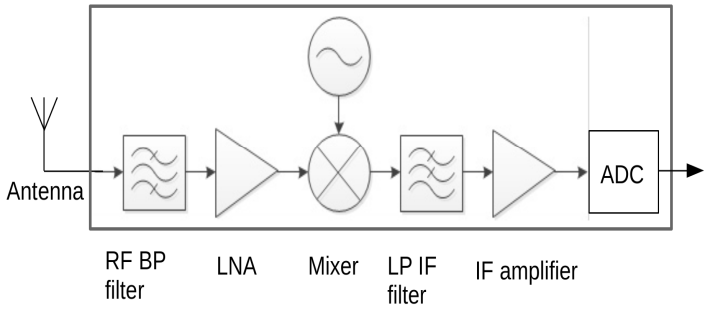


Fig 3.2 Funcube Dongle Pro + schematic.

This other dongle will be receiving at the 40 meters band because there is always some hams activity, and is a frequency that the other dongle, which is used on the project, is unavailable to achieve. So, it's more interesting to use this other receiver in order to complement the cadences of the RTL-SDR one (talking about the frequencies that each one can achieve).

3.2 Antennas

The antenna is an important part of the system, perhaps this can be the most basic part of the system. In this system, the antenna should receive all the signals on the desired bands with the better quality possible. In this part, no

antenna comparison will be found, just the chosen antennas and the general reasons why the model was chosen and its assembly.

But before start with the antennas, is so important to understand **the reason why the two frequency bands where chosen.** For that reason, this explanation must came first, just before start with the antennas.

3.2.1 Frequencies justification [45]

3.2.1.1 2m band (144MHz)

The first band that we choose was the **2m band** (144MHz) in the RTL-SDR. The frequencies that this dongle can achieve are from 24MHz to 1766MHz in this range the ham bands are 12m, 10m, 6m, 4m, 2m, 70cm and 23cm (more information about the ham bands can be found on the annexes). In this bands, the more actives (the ones that are more used by the hams) are the 2m for VHF and 70cm for UHF bands, but the most used and the most interesting for us is the 2m one, because it has more propagation and we can find more users.

This band is so interesting because of its multiple and variate usages, from the direct connections and the repeaters activity to some spacial communications or satellites reception. An interesting data, is that the ISS (International Space Station) uses the 145,800MHz frequency modulated in FM in order to stablish contacts with the hams around the world.

3.2.1.2 40m band (7MHz)

The dongle that we set in this frequency is the Funcube Pro +, that has a frequency range from 150 kHz to 240MHz and 420MHz to 1.9GHz. In this frequencies, is so important to see that we have all the lower frequencies that the first dongle cannot achieve. So, it's much more interesting if we center our efforts in choose the better band in that lower frequencies range, in order to use this advantage.

In the lower frequencies range we have the 630m, 160m, 80m, 60m, 40m, 20m, 17m, 15m and some more that we can get from the RTL-SDR, so we will not take them into account from this set of frequencies. We can see that if the idea is to deploy an antenna, the first two bands even if we build a quarter wave length antenna will be huge, so they are discarded.

For the other ones, the best option is the 40m band, because is the most active one here in Spain, as all the HF bands are usually used for ionospheric propagation. So, we can achieve high distances, depending on some factors like the temperature, the solar activity, … But this specific band, is interesting because early in the morning, when the propagation is not in good conditions, we can hear near transmissions from Spain, France, Portugal or Italy, because of the bad propagation conditions. It is difficult to establish a high distance communication, but in the afternoon when the propagation conditions start to

become better further distances communications can be stablished, what makes this specific band extremely active.

Another reason for this band to be extremely active, is the ludic part of the hams, there is a range of frequencies that goes from 7130 KHz to 7200 KHz, that is one of preferred ones in order to make contests.

3.2.2 2m Band antenna

The half wave antennas generally occupy too much space as shown on fig 3.3 and are difficult to lift. Knowing that, when the ground is a good conductor, it acts as a reflector as shown on fig 3.6. This make possible to have an antenna that have the half of the size, but the same length from the electrical point of view. Nevertheless, the ground conductivity changes on each location and also is subject to atmospheric conditions. As a conclusion of this, is a better option to build an artificial ground which conductivity and reflector properties stays invariable and determined.

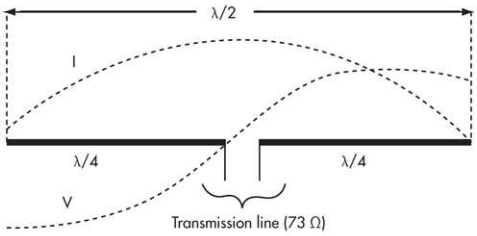


Fig 3.3 The patterns show the distribution and strength of the voltage (V) and current (I) on a dipole antenna. [41]

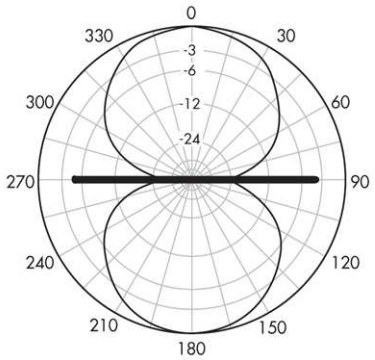


Fig 3.4 Horizontal (Electrical field) radiation pattern of a half-wave dipole antenna. (The antenna elements are located along the 270° to 90° line.)[41]

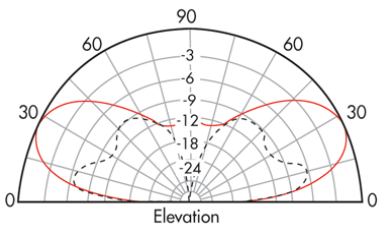


Fig 3.5 Vertical radiation pattern showing elevation of signal from a horizontal dipole one half wavelength above the ground. [42]

We can build this artificial ground by adding 4 conductors on the base of the vertical antenna shown in figure 3.6, and making them have an angle between each one of 90º. This conductors must have a minimum length of a quarter of the wavelength.

Is also important to know the angle between the ground plane and the “active” element, this is because the angle that exists between both will affect to the antenna impedance.

This antenna electrically does not represent any advantage respect to a dipole but physically we can reduce the occupied space maintaining the omni-directional radiation pattern.

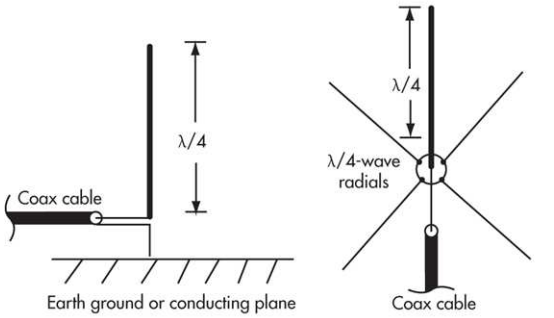


Fig 3.6 Ground plane monopole is one quarter wavelength high. The ground or radials are the other half of the antenna. [42]

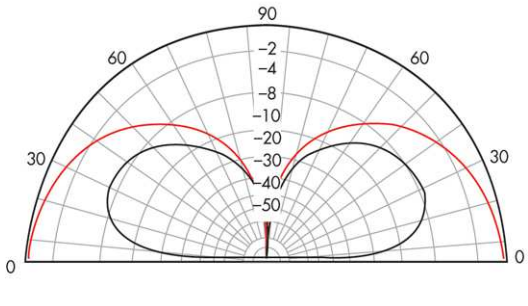


Fig 3.7 This is the vertical radiation pattern of a ground plane antenna over a so-called perfect ground. [42]

3.2.2.1 Implementation

In order to implement this antenna, the only need that someone can have is the connector and the conductor material.

In our case, we choose a PL square connector with 4 holes at the corners in order to make easier to connect the ground plane conductors, as the chosen band is 2m the quarter of wavelength (λ) will be 50 cm as is shown on the equation (3.3). One thing on this equation can be confusing, that is that the frequency is not 144MHz is 145MHz this is not a mistake, this is because the 2m band takes from 144MHz to 146MHz. So, in order to have an antenna that can receive the signals from the entire band, we must calculate its dimensions taking into account the central frequency.

$$\lambda = \left(\frac{3 \cdot 10^{8}\ [\mathrm{m/s}]}{145 \cdot 10^{6}\ [1/\mathrm{s}]} \right) = 2{,}0689m \qquad \textbf{(3.3)}$$

Fig 3.8 Used PL Connector

For the conducting section copper pieces where used, this pieces have diameter of 3mm and are not any copper alloy, is pure copper.

Fig 3.9 Copper pieces

In order to assembly the antenna, the active piece of copper was cut several centimeters longer in order to adjust the resonance frequency. The conductor was cut until the maximum resonance frequency match with the center frequency of the desired band.

Also as is explained before, the angle between the ground plane and the active element is something to take into account and it has to be 135º in order to achieve a impedance of 50 ohms. If the ground plane and the active element has an angle of 90º, the antenna will have a 75 ohms impedance. This will be a problem because of the no adaptation to the line that takes the signal to our dongles.

In order to assembly the ground plane, we need to preheat the surface of the PL square connector because its composition makes very difficult to heat this surface. For that reason, a hot air blower was used to preheat the connector heating with hot air at a temperature of 375ºC. Then, the soldering became easier for the ground plane.

Fig 3.10 Air Blower.

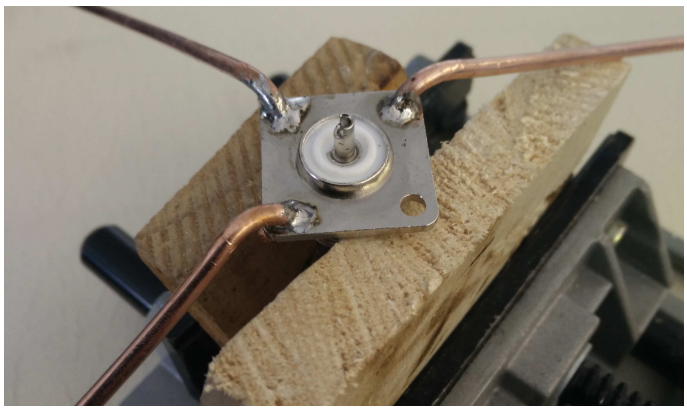


Fig 3.11 Connector with some of the copper pieces that forms the ground plane.

For the antenna calibration, the procedure was to connect the antenna to a spectrum analyzer and checking the S11 parameter in the log scale vs frequency. We left some extra centimeters on the active element in order to

adjust it manually (not from calculus), so on the first moment the resonating frequency was lower than we want as shown on the fig 3.14.

Fig 3.12 Calibration Scenario.

Fig 3.13 Adjust Procedure.

Cutting the copper, we will make the antenna adapted for a different frequency, displacing the resonance frequency to a higher frequency.

After the calibration, the S11 parameter was the desired being the center of the antenna resonance at 145 MHz, as is shown on Fig 3.15.

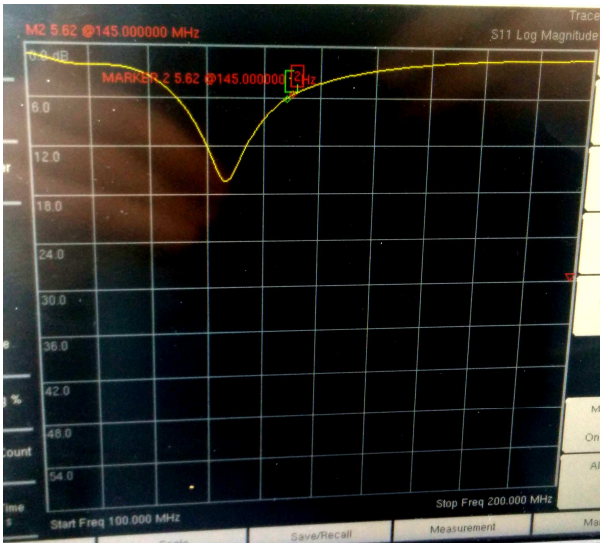


Fig 3.14 First S11 measurement dB vs freq, note that in the image appears 2 markers the green one represents 144MHz and the red one 146MHz so the resonance frequency at the end of the calibration must be in the middle of the two points.

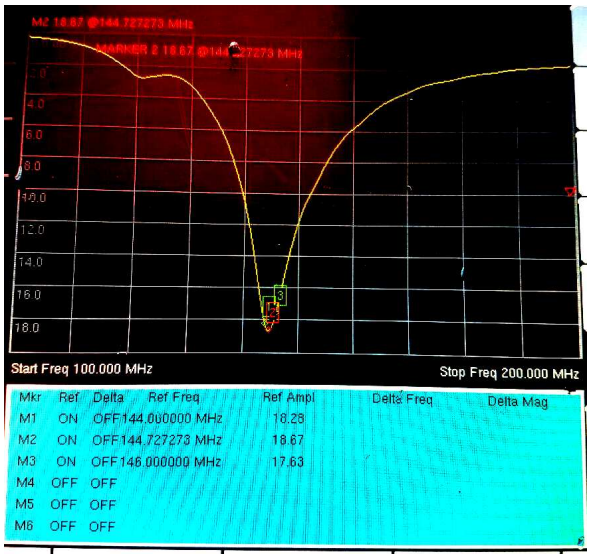


Fig 3.15 Final S11 measurement dB vs freq, note the three markers frequencies (144 MHZ, 145MHz 146MHz).

Other interesting parameters to measure are the voltage standing wave ratio (VSWR) and the impedance of the antenna on the smith chart.

The VSWR is an important parameter because even if we have the resonance frequency on the desired range that does not guarantee to us that this antenna has a good performance. It is also important to know its impedance and VSWR in order to know if we have a good adaptation to the rest of the circuit. A bad adaptation will make the system to perceive a lot of reflected waves during transmission. As a consequence, this reflected waves power is great enough to damage the transmitter terminals. This is the reason why the VSWR values must be checked on the antenna.

In order to calculate the VSWR, a Spectrum analyzer was used. Aiming at understanding the concept, the VSWR will be defined with a few simple equations:

$$\Gamma = \left(\frac{Vr}{Vf}\right) = \frac{Zl - Zo}{Zl + Zo} \qquad \textbf{(3.4)}$$

The equation (3.4) shows the definition of the reflection coefficient, which is a complex number that describes the magnitude and phase shift of the reflection. Its definition, as shown in the equation, is the relation between the forward wave Vf and the reflected wave Vr. We can also define it from the impedance relations between the antenna and the coaxial that will be connected.

This coefficient will take values between -1 and 1, which means that if is -1, 100% of the wave will be reflected and no power will be transmitted to the

antenna. As a results, the reflected wave will cancel the forward wave and probably damage the equipment.
If the coefficient is 1, it means that all the signal is reflected again, but this time the reflected wave will be on phase with the forward signal.

The ideal value of the coefficient is 0, it means that the amplitude of the wave remains constant. From the impedance point of view, the antenna has the same characteristics as the coaxial impedance. Therefore the circuit is adapted and all the power will be transmitted to the antenna.

If we think about how the reflected wave will affect our antenna. This is easy to understand, the antenna will have a wave that goes forward and another with less power that goes backward. It means that in a determined point, the wave will be the sum of all the waves in that specific location.

That means that the maximum amplitude that we will have for any point of the system is the sum of the transmitted and the reflected power. The minimum amplitude will be when the forward and backward waves are in counter phase, which is the same as subtract the value of the reflected and forward waves. With all this information, now is easy to see that the VSWR is defined as it shows on the equations (3.5), (3.6) and (3.7).

$$|V\max| = Vf + Vr = |Vf| + |\Gamma \cdot Vf| = (1+\Gamma)\cdot|Vf| \qquad \textbf{(3.5)}$$

$$|V\min| = Vf - Vr = |Vf| - |\Gamma \cdot Vf| = (1-\Gamma)\cdot|Vf| \qquad \textbf{(3.6)}$$

$$VSWR = \frac{|V\max|}{|V\min|} = \frac{1+|\Gamma|}{1-|\Gamma|} \qquad \textbf{(3.7)}$$

On the figure 3.16, the VSWR was measured with a spectrum analyzer and setting 3 markers at the most interesting frequencies for us, that are: 144 MHz for being the starting frequency of the desired band, 145MHz (on the Spectrum Analyzer we can just put a marker on 144,7272 MHz not exactly on 145 MHz). For being the center of this desired band and 146MHz for being the last frequency of this band. The results were quite good being the VSWR for the three mentioned frequencies: 1,07:1 at the first one, 1,14:1 on the central and 1,27 on the last.

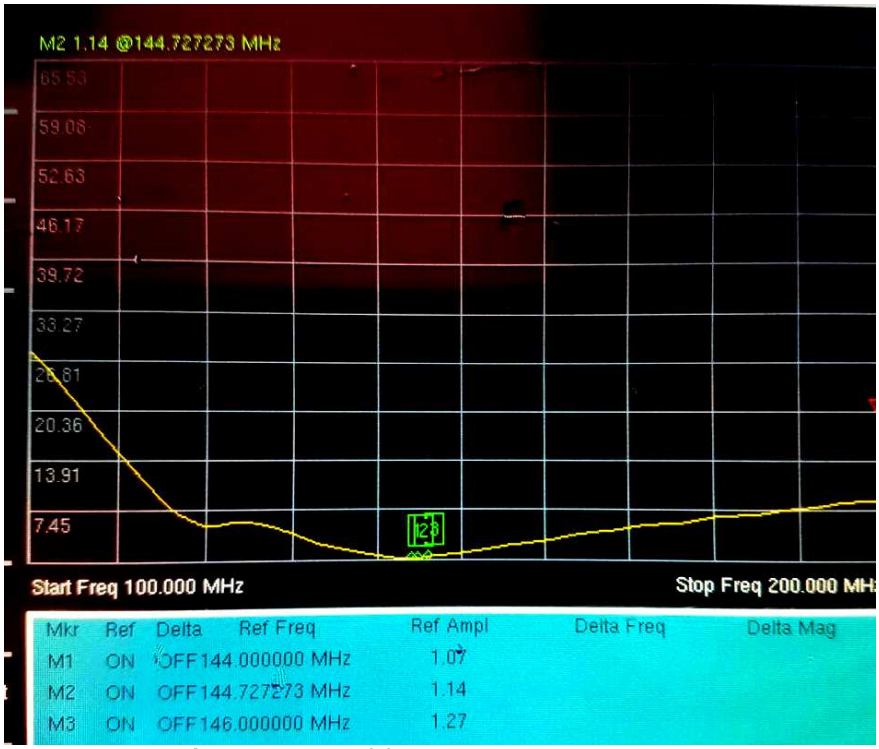


Fig 3.16 VSWR of the 2m atenna

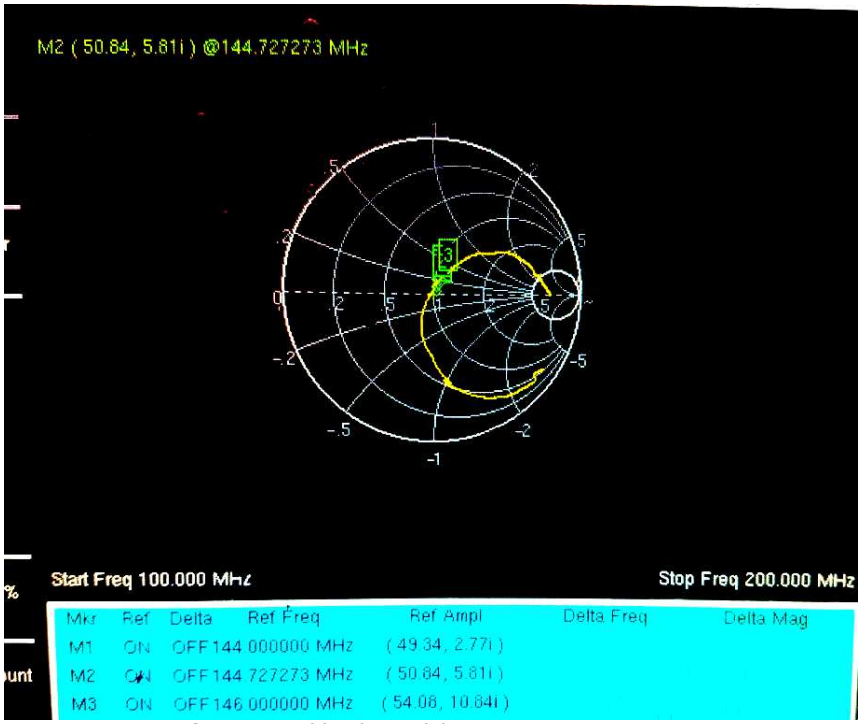


Fig 3.17 Smith Chart of the 2m antenna

As we can see on fig.3.17, the antenna has a good impedance on all the band being worst at the last frequency of the band. But in any case, the imaginary part of the impedance always is not a problem having a maximum VSWR of 1.27:1 on the last frequency of the band, which corresponds to the worst adaptation of the antenna in our band.
In order to receive the maximum possible signals we will put this antenna with a vertical polarization.

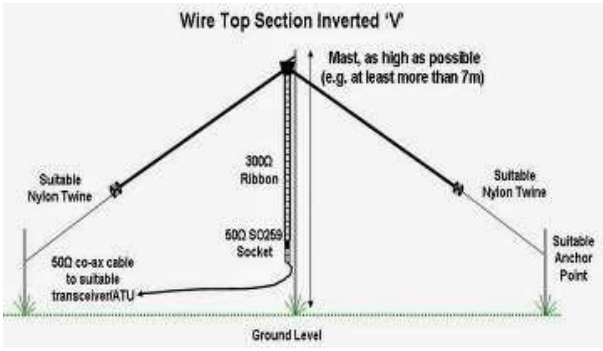
3.2.3 40m Band antenna

The dipole antenna can seem the perfect antenna for a specific frequency. It is mostly true, it's a simple structure, if this dipole is for some band reception, just the center frequency of the band should be known and then it is a really simple configuration as shown on fig 3.3. The antenna will be assembled from 2 pieces of wire with a length of a quarter of its wavelength, and if all works well and the band is narrow enough probably we will have a low enough standing wave ratio (SWR) on all the frequency range inside the band.

The dipoles have a narrow bandwidth due the simple 2 wire design and there are no loading coils and no DC ground. They are also subject to noise and static, but there is another type of antenna that can solve this little problems.

Double Bazooka is a unique design alternative to a dipole, it is made of coax cable, instead of a copper wire, and is the braided copper shield of the coax the RF radiator, meanwhile the central conductor acts as a balun, to provide ground.
The larger diameter of the shield braid gives the bazooka a wider range of frequencies on each band with lower SWR. There are some possible configurations of this antenna position, the 2 most used are:

-Extended dipole
-Inverted vee



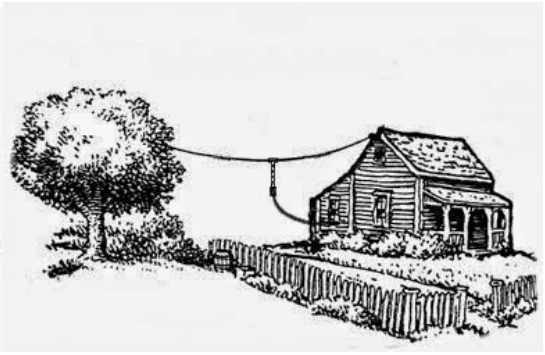


Fig 3.18 Inverted vee configuration [42]

Fig 3.19 Extended dipole configuration [42]

Each configuration has his advantages and withdraws, but the most interesting for us is the inverted vee, which is in this case, the best option making each of the sides in order to reach an angle with the horizontal of 45º, what makes the antenna capable to receive vertical and horizontal polarizations easily.

The double bazooka antenna is extremely silent, meaning that has a really low WSR and good for wide range bands. It has some more gain that a half wave dipole what makes it ideal for the desired 40 m band.

To assembly this antenna, we used RG 213 wire for the section A of the fig.3.20 and a parallel line of 300 ohms as a short-circuited stub (section B of fig.3.20).

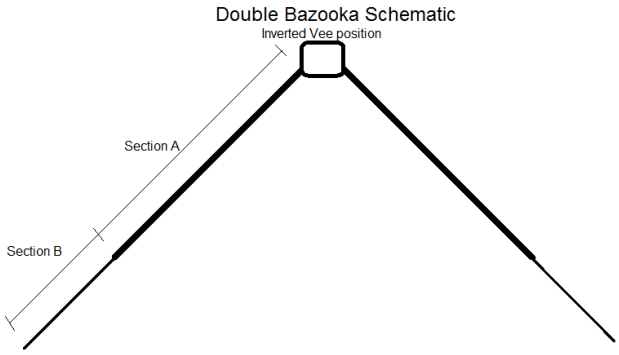


Fig 3.20 Inverted vee double bazooka schematic

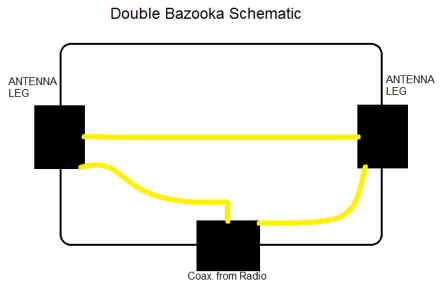


Fig 3.21 Inner box detail

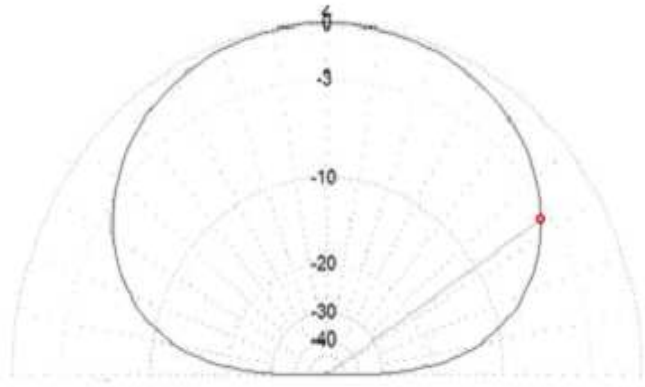


Fig 3.22 Horizontal radiation pattern of a double Bazooka antenna on inverted vee position. [56]

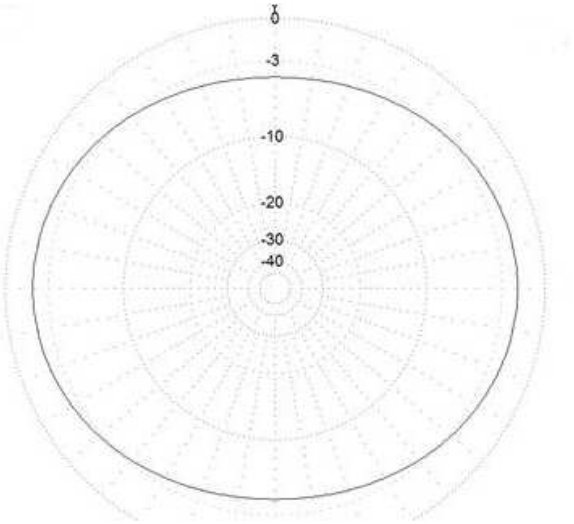


Fig 3.23 Vertical radiation pattern of a double bazooka antenna in an inverted vee position. [56]

Implementing the antenna as happened with the other one we left a few centimeters more in order to adjust the length for the desired frequency empirically later, the procedure for adjusting was the same, cut both extremes and short-circuit the extremes of the stubs gain.

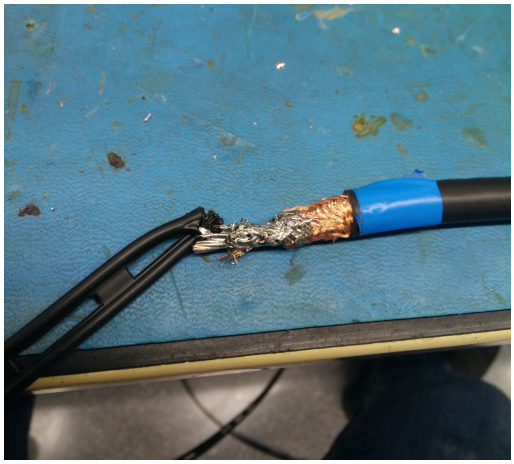


Fig 3.24 Detail of the connection between section A and B

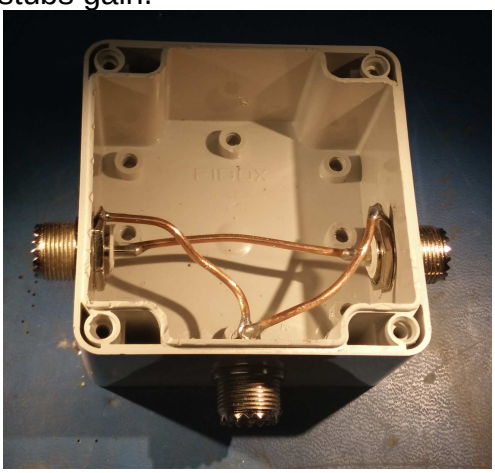


Fig 3.25 Implementation of the connection box

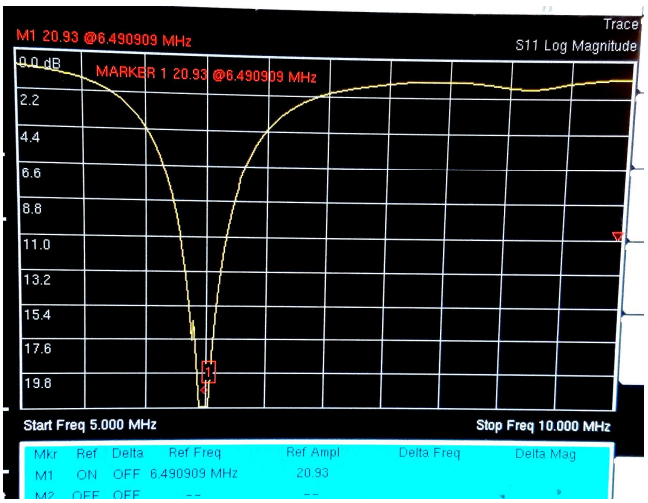


Fig 3.26 First S11 measurement dB vs freq, on the graphic the marker is set at the minimum of the peak that is centered at 6,490909 MHz with a ref ampl of 20,93 dB .

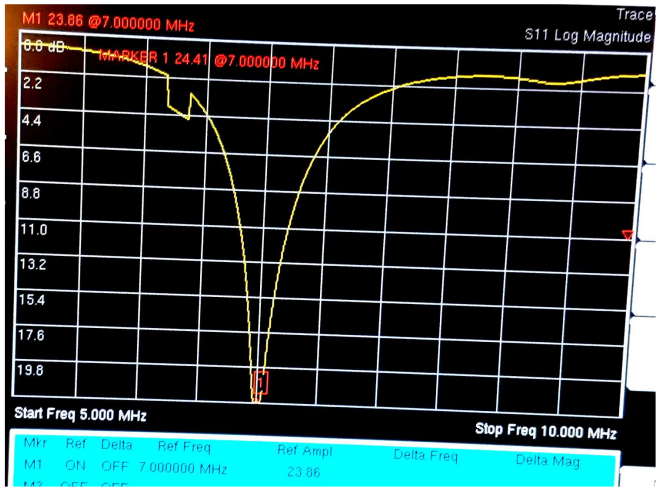


Fig 3.27 Final S11 measurement dB vs freq, on the graphic the marker is set at the minimum of the peak that is centered at 7,00 MHz with a ref ampl of 23,85 dB .

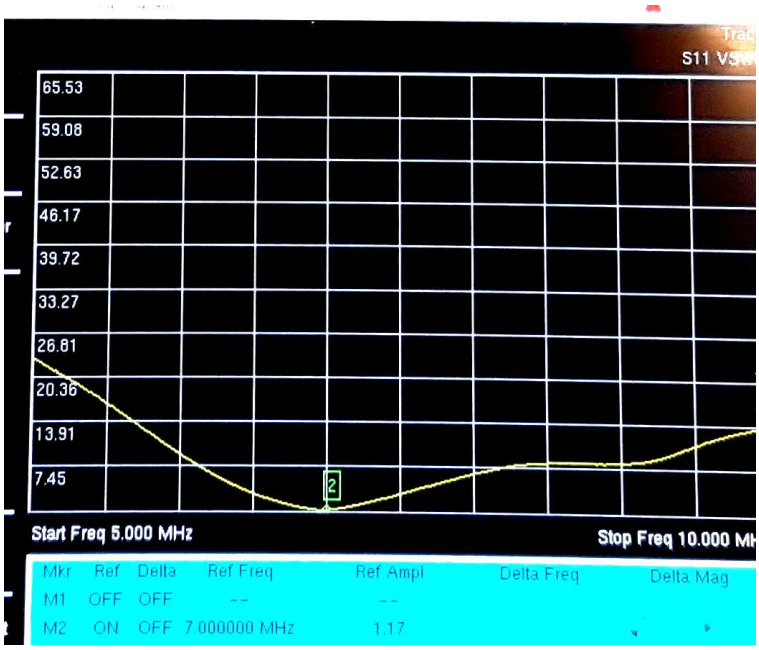


Fig 3.28 VSWR measurement at 7MHz

The final resonance frequency was 7,0000MHz exactly, what is quite good results that later we can see that the antenna have a very good adaption with the VSWR value that is 1,17:1. This is almost 1:1 what is the perfect value because it means the antenna impedance is the same as the reference impedance given by the coaxial.

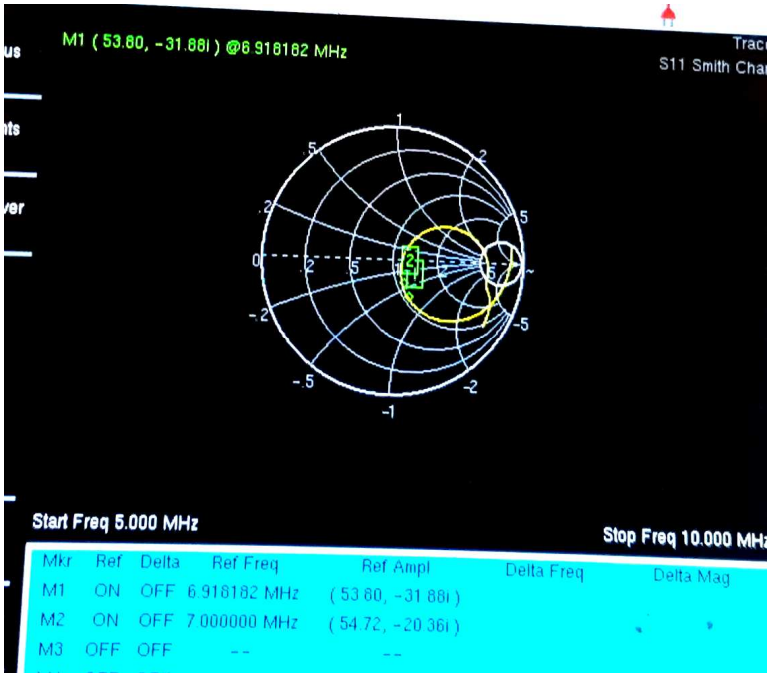


Fig 3.29 Smith Chart from 5MHz to 10MHz using a marker at 7MHz.

The double bazooka antenna has a wider frequency range than a normal dipole, we can check the bandwidth by considering a good adaptation a VSWR, as example the most of the commercial antennas on the datasheet allows a VSWR until 1,5. If we take a limit of 1,5:1, we can check on the fig 3.28, we can see approximately that we have 1MHz for this antenna after that the return losses will be lower than the limit that we set, which means that more energy will came back to the transceiver what can be a problem during a transmission, on reception also we will have a problem because the minimum level of a signal to be detected will also increase, so some weak signals will be lost because of a wrong adaptation.

3.3 Software

The hardware that composes this system has been explained but it remains the other main part of the system which is the software that makes this project work. That is what this section is about all the software decisions will be explained and justified.

3.3.1 OpenWebRX based

From the software analysis that was done on the SDR and webSDR sections, we find out that are lots of webSDR software already implemented, so it has no sense to reinvent the wheel, so we centered on find out the best option for the proposal that this project has taking into account that this webSDR will have also educational purposes and will be object of future studies and improvements. For that reason, one of the most important considerations was to use a free software, because this opens the possibility of change and modify the software when is required.

The OpenWebRX seems the best option due its initial purpose that is quite similar to the one of this project, that was to proportionate to the hams a good tool in order to hear the signals without the need of having a receiver themselves and also for educational purpose.

This software is super interesting and complex at the same time. We are going to explain in the better way that will be possible how it works. To start understanding this, we can take a look to the fig.3.8, on that schematic we can roughly say that there are 4 main blocks or processes which are:

rtl_thread: This thread connects with the dongle in order to obtain the I+Q raw data and sends it to the rtl_mus_thread.

rtl_mus_thread: This thread acts as a hub it is used for distributing the I+Q data between all the process. It acts as a TCP server to stream the raw I+Q data to the multiple local functions that demands it.

spectrum_thread: This other function requires his own thread that must connect to the rtl_mus_thread in order to obtain the raw I+Q data and process it in order to obtain an array of dB values that will be unique for all clients, this array of dB values is the one that will be used on the client side to create the warterfall view.

httpd: Is a thread that performs the most “interesting” functions for the receiver it connects to the DSP module in order to process the client requirements (modulations, squelch, …) also connects with the rxws module that receive all the client messages and process them, stablishes the socket connection and other interesting functions. A new instance of this thread is created each time that a client connects to the server in order to handle its petitions or requests.

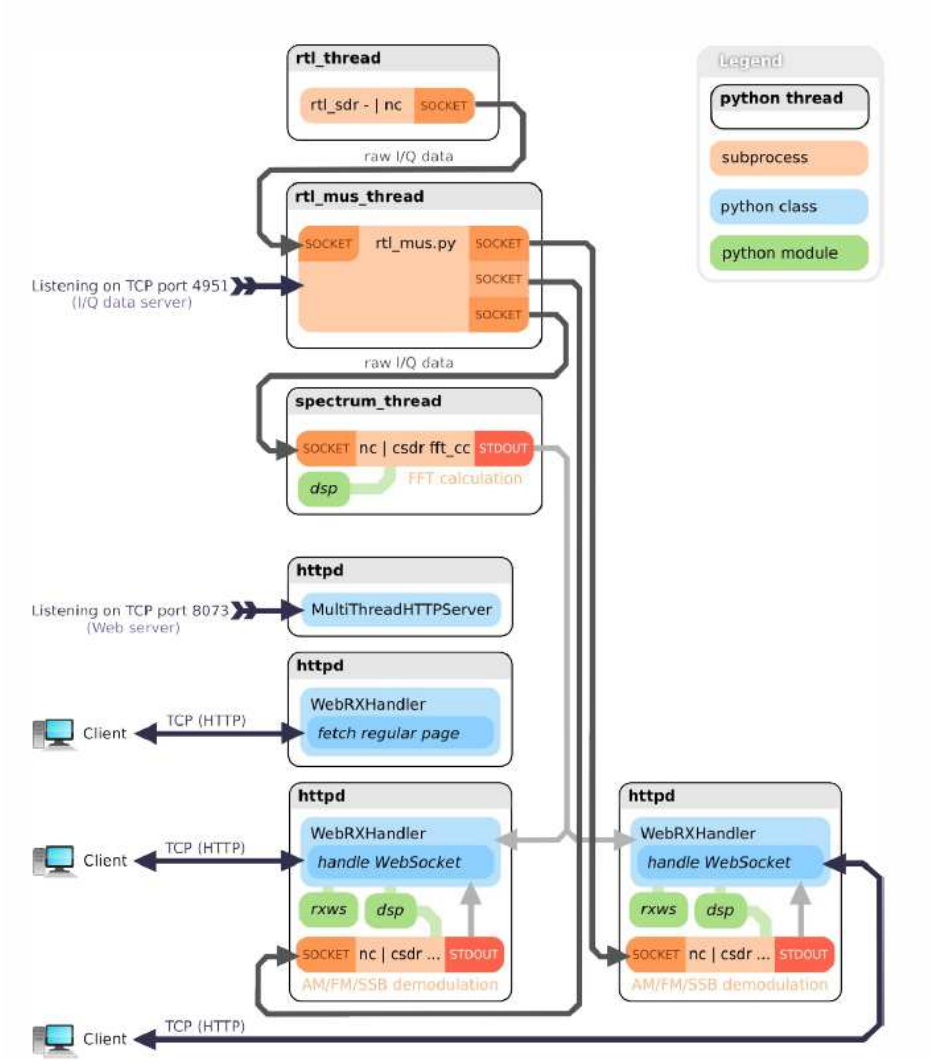


Fig 3.30 Simplified diagram of the OpenWebRX modules [47]

It is interesting to know that the DSP python function is not the DSP software, itself it is a class in python that sends the instructions to the DSP library that is on the server that is written in C to handle the real time requirements of the system.

Also the client is not shown on this schematic, the client is a HTML5 webpage that uses JavaScript (JS) as an engine in order to be interactive. This JS functions are written in two JS files which are:

Openwebrx.js: This contains the control of the UI and the web socket, draws the waterfall making an assignation of the colors for a corresponding power received in dB, which is sent from the server httpd thread, which forwards this information when it receives the data from the spectrum_thread, and also outputs the sound using Web Audio API.
Sending the dB power of each division of the frequency band of the spectrum and audio, can consume a high bandwidth on the connection. For that reason, the audio is down sampled until 11025Hz and later is compressed using adaptive differential pulse code modulation (ADPCM), the spectrum data also is compressed with ADPCM before sending to the client, that allows to reduce the

bandwidth required for the client until 200Kbits/s, which are 70Kbits/s for the audio and 130Kbits/s for the spectrum data.

sdr.js: This other file contains some functions for audio processing that is required when is received from the server before sending to the client audio.

In order to test the system capacity (clients) we run some tests using the RTL-SDR dongle and varying the sampling rate of this receiver in order to increase or decrease the bandwidth. For the sampling rate of 250 kHz the bandwidth (BW) was of 0,22MHz for a sampling rate of 1024 kHz, the BW was 1,22MHz for 2048 kHz the BW was 2MHz and for a sampling rate of 2400 kHz the BW was 2,4MHz.

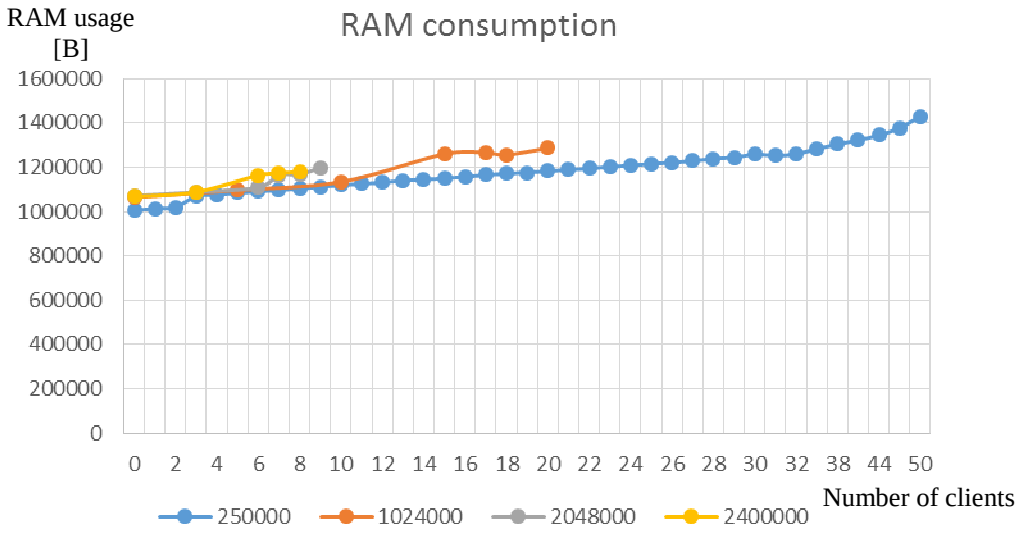


Fig 3.31 Server RAM consumption vs number of clients for 4 different sampling rate values

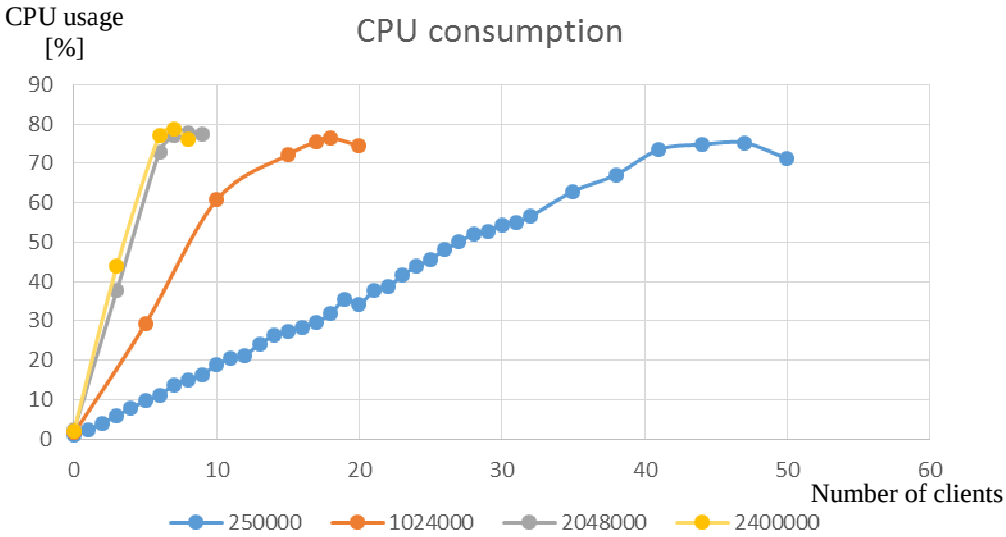


Fig 3.32 Server CPU consumption vs number of clients for 4 different sampling rate values

Knowing that the total amount of the PC RAM memory is 2GB the server can waste the 75% of it what means that we can saturate the PC in order to avoid that we must limit the number of connections to the server. Also use 80% of a

3GHz processor is a quite high number for just a few clients but we can see that process all the data for a high bandwidth can be a computational problem.

As is shown on the fig3.21 and 3.22, for a higher samplerate and for instance, a higher bandwidth (more processed frequencies) less clients can be achieved.

That is because for a higher bandwidth more data of the entire band has to be sent through the connection and also the server have to decode much more information for each different user. For that reason, the saturation point is reached earlier.

Based on this results, the chosen sampling frequency was 2048kHz because even if not more than ten simultaneous users can be connected all the band can be seen at same time. As the system is reconfigurable, in any moment if the users are increasing for above of 10, we can allways decrease the BW to 1,22MHz in order to allow almost 20 simultaneous users.

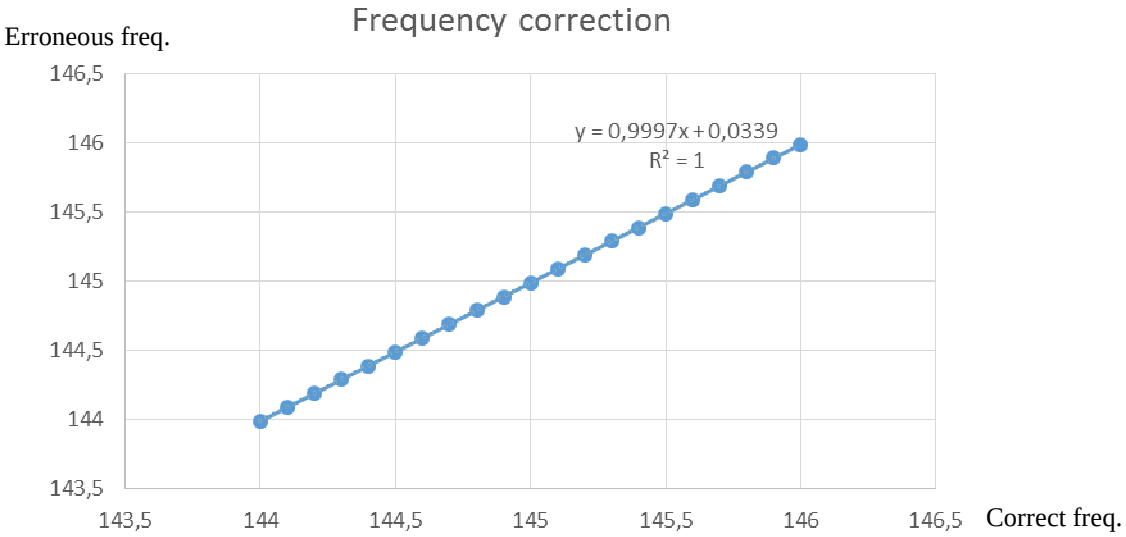


Fig 3.33 Dongle frequency correction

During the dongle tests we realized that the RTL-SDR has a small frequency deviation (the frequency displayed was a bit higher than it should be), so we have checked its error and we have corrected in order to receive a more accurate frequency information on the web.

The correction procedure is simple, we have to make the inverse process at the time that we assign the frequency value as shown on equations (3.9) and is proved on the equation (3.10).

$$Error\_freq = 0{,}9997 \cdot Correct\_freq + 0{,}0339 \qquad \textbf{(3.8)}$$

$$Correction = \frac{Correct\_freq - 0{,}0339}{0{,}9997} \qquad \textbf{(3.9)}$$

$$Correct\_freq = 0{,}9997 \cdot \left(\frac{Correct\_freq - 0{,}0339}{0{,}9997}\right) + 0{,}0339 = Correct - freq \quad \textbf{(3.10)}$$

This correction is applied on the configuration server script as:

center_freq_corrected = int(round((center_freq-0.0339)/0.9997))

For the Funcube Pro + dongle we don't have this possibility of changing the span, because it's sampling frequency is fix at 192KHz it means that we can just support around 60 users. So we can try to share the PC services between the two dongles just limiting the connections to 5 or 6 on the RTL-SDR and to 30 into the Funcube Dongle.

3.3.2 Additional software

In order to make the whole system work, we need some extra libraries or software that is not embedded in OpenWebRX.

In order to explain all extra software that is needed we can take a look to the figure 3.21.

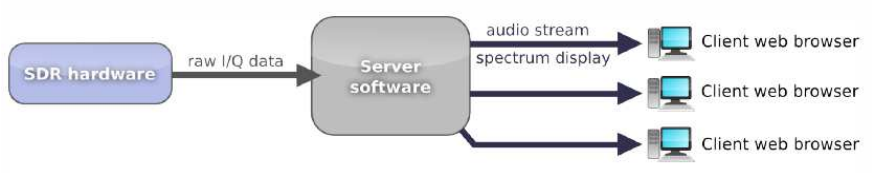


Fig 3.34 Simplified diagram of the system modules [47]

In this figure, we can see the main needs of the system which are basically a program that allows the SDR to communicate with the server. As we said before, some extra libraries that will be placed on the block of server software but is not embedded on the OpenWebRX software. Also a demodulator for BPSK31 was implemented as a prototype on Scilab, as we will explain later.

3.3.2.1 Dongle connection with server

In order to allow the server to receive the I+Q raw data from the dongle, we should install some software that allows to set the center frequency and the sampling rate and other essential parameters to the dongle. In order to do this function, the following software has been installed on the server:

Rtl-sdr [48] is an Osmocom software that allow us to control the **RTL-SDR** allowing some parameters like the center frequency or sampling rate, the command that we must use to set the parameters to the dongle should look something like:

rtl_sdr -s [samp_rate] -f [center_freq] -p [ppm] -g [rf_gain] –

For the **Fun Cube Pro +** dongle we use **QTHID** which is a simple controller specifically for this dongle and has a graphical interface that allows the modification of the parameters in real time as shown on fig.3.35.

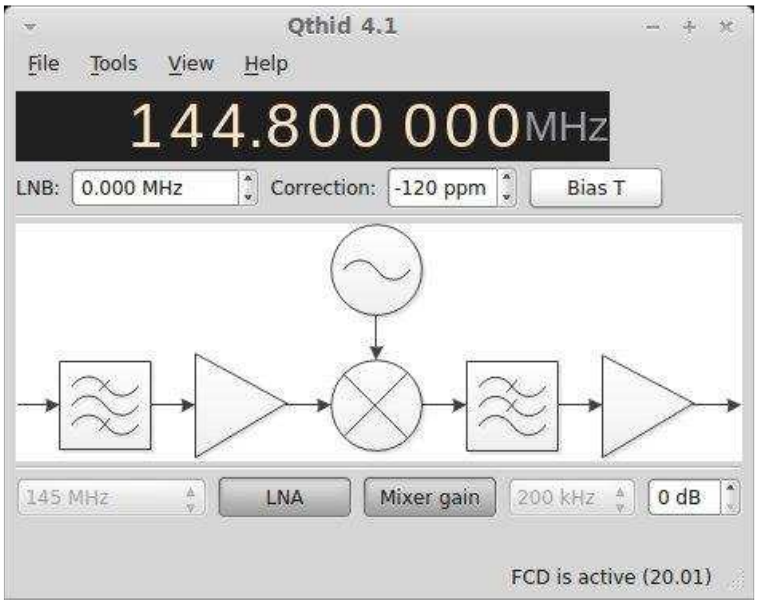


Fig 3.35 QTHID GUI view [48]

3.3.2.2 DSP library

As is explained before, the OpenWebRX software needs a DSP library external in order to process all the signals. This library is written in C in order to have a real time processing. The OpenWebRx software just have an array of commands to send to the DSP library, in order to process the data as the client expect.

To have an example of what it means, we can check a part of the OpenWebRx code that communicates with the DSP library:

if which == "nfm": return chain_begin + "csdr fmdemod_quadri_cf | csdr limit_ff | csdr fractional_decimator_ff {last_decimation} | csdr deemphasis_nfm_ff 11025 | csdr fastagc_ff 1024 | csdr convert_f_s16"+chain_end

elif which == "am": return chain_begin + "csdr amdemod_cf | csdr fastdcblock_ff | csdr fractional_decimator_ff {last_decimation} | csdr agc_ff | csdr limit_ff | csdr convert_f_s16"+chain_end

elif which == "ssb": return chain_begin + "csdr realpart_cf | csdr fractional_decimator_ff {last_decimation} | csdr agc_ff | csdr limit_ff | csdr convert_f_s16"+chain_end

This library is as deduced from the code above csdr, wich contains a set of simple DSP routines for SDR, and the most useful ones are the demodulation of AM/FM/SSB and the spectrum display.

3.4 BPSK31 demodulation code

As is explained on the introduction of this chapter, a prototype of BPSK31 demodulator has been written on Scilab. This is a complement to this project that was thought during the software analysis, we realized that all the SDR programs (not WebSDR) have some digital de/modulators implemented. Meanwhile, the WebSDR software was mainly centered on the analog modulations.

So it was an interesting line to start implementing a digital de/modulator to give additional value to this project. The decision was to center the efforts on understanding and decoding the most commonly used digital modulation, and this is BPSK31.

3.4.1 Introduction

PSK31 codification uses a varicode, this is a Huffman code applied to the characters, which means that the most used characters will have a shorter codification in order to improve the channel efficiency. Getting in this way, the minimum time usage for each letter which means that we will transmit more data in the same time than if we use a traditional mapping in which we assign the same amount of bits. For each symbol, we can see in the figure 3.36, the comparison between the transmission of the code “ten” codified in ASCII, RTTY, Morse and varicode.

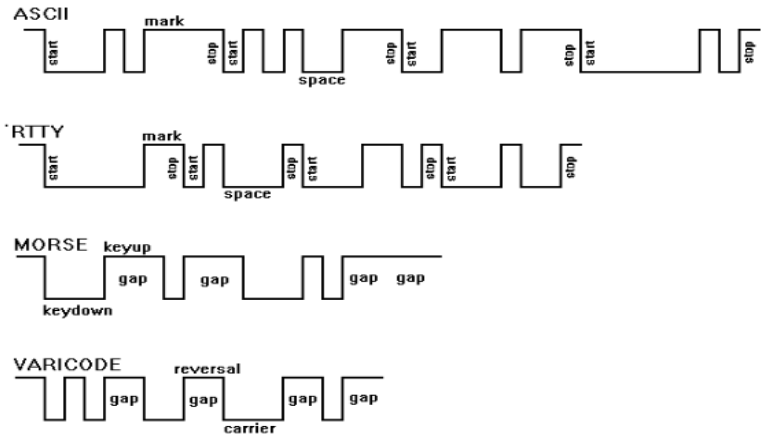


Fig. 3.36. Comparison between ASCII, RTTY, Morse and varicode. [50]

In order to transmit this modulation we can do it in two ways:
-BPSK31
-QPSK31

Obviously, as the name tells the main difference, this is the transmission mode that can be Binary Phase-shift keying (**BPSK**) or Quadrature Phase Shift Keying (**QPSK**), for instance the bps that we transmit are also different for both of the modes.

This prototype will center the efforts in BPSK31, because is the most used by ham. This does not mean that they don't use QPSK but is not common. The main characteristics of this type of modulation are found on the recommendation ITU-R M.2034.

The most interesting ones for transmission and reception are the **baud rate** and the **characters mapping.**

From the document, we can find that the baud rate is 31,25 what is approximately equivalent to say that we can transmit 50 words per minute.
From the baud rate we can also obtain the symbol period, knowing that the baud rate are the symbols per second we get a symbol period of: 0,032s

The most ordinary frequencies to transmit in PSK31 are shown on Table 3.5.

Table 3.5 Compilation of the most used frequencies to transmit BPSK31. [51]

Frequency	Band
1.838 MHz	160 meters
3.580 MHz	80 meters
7.035 MHz	40 meters (region 3)
7.070 MHz	40 meters (regions 1 and 2)
10.140 MHz	30 meters
14.070 MHz	20 meters
18.100 MHz	17 meters
21.080 MHz	15 meters (most activity found 10 kHz lower)
24.920 MHz	12 meters
28.120 MHz	10 meters
50.290 MHz	6 meters

3.4.2 Decoding BPSK31

The first step was to check which can be the signal that we receive, knowing that our OpenWebRX software can filter the signal that we want avoiding all the rest of signals that can interfere with the desired signal, we know that we will receive our signal "clear". In this case, clear means that we will receive just one signal with some noise, but only one signal, the user must graphically filter the signal as accurate as possible.

Knowing this and as was explained before, we need to know our BPSK signal to be decoded, we can take a look to a BPSK31 signal to try to understand the steps.

On the fig 3.37, we can see that a preamble sync sequence is sent, but on a server, you never know if you are going to start hearing at the start or in the middle of the transmission so is useless.

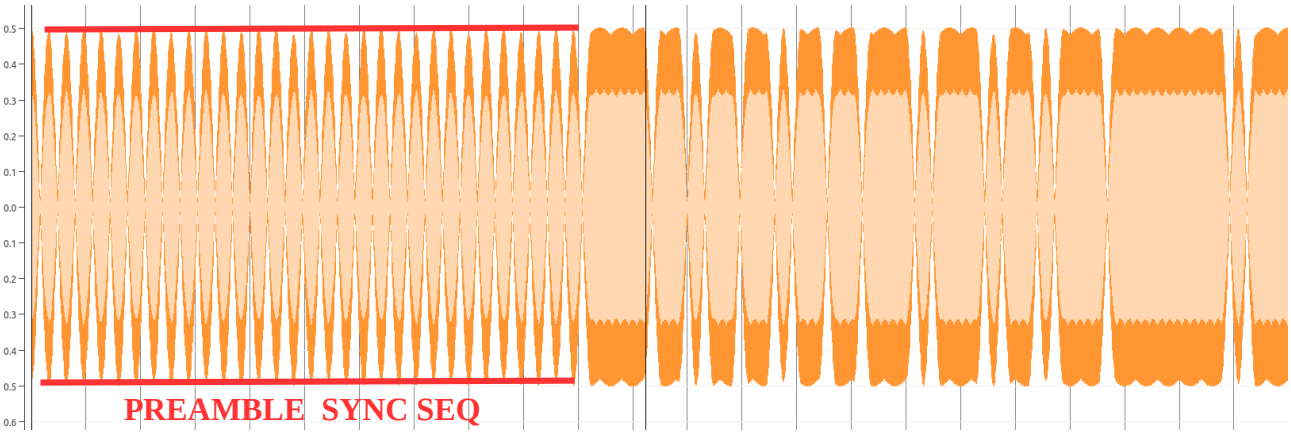


Fig. 3.37. BPSK31 signal on time domain

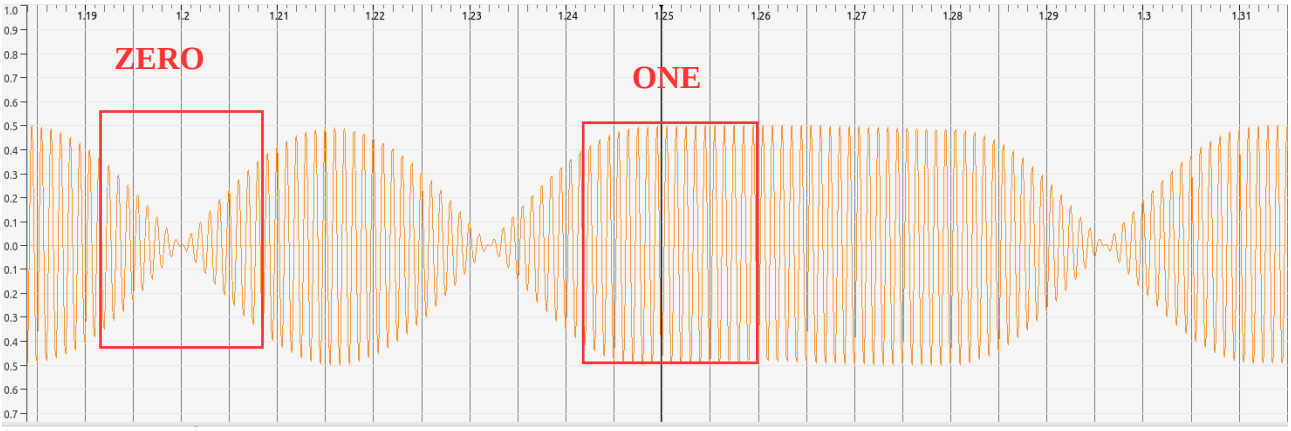


Fig. 3.38. BPSK31 signal on time domain zoomed to allow see the differences between ‘1’ and ‘0’

We can see from the caption fig.3.39 that the BPSK31 is a narrow-band signal which is quite good for transmission when we have a small part of spectrum to use.

In order to decode the signal was implemented the following procedure in Scilab to test.

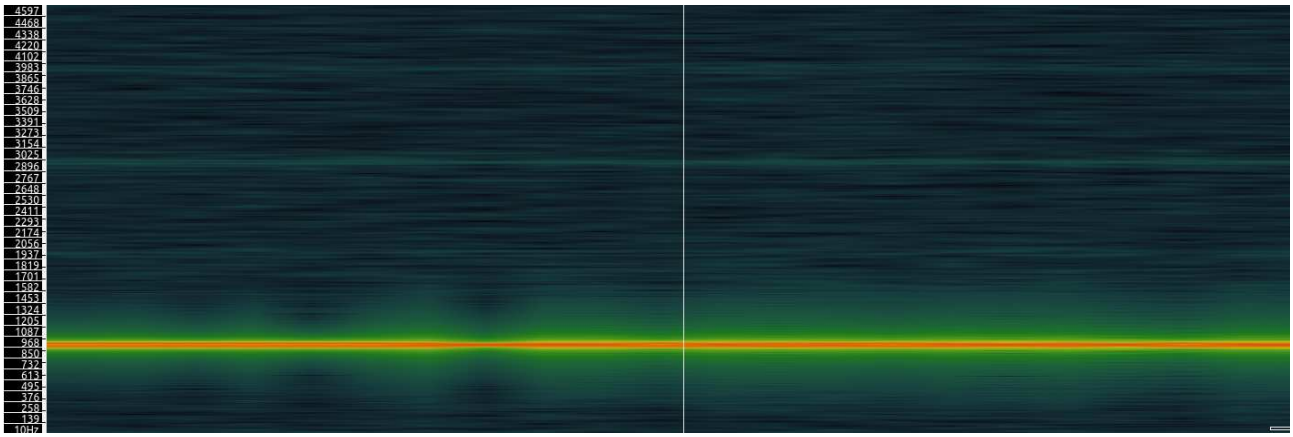


Fig. 3.39. BPSK31 signal on horizontal waterfall spectrum

For explaining the developed code a graphical example will be used:

A test vector (set of samples with a known message) was used, this is shown on Fig3.40.

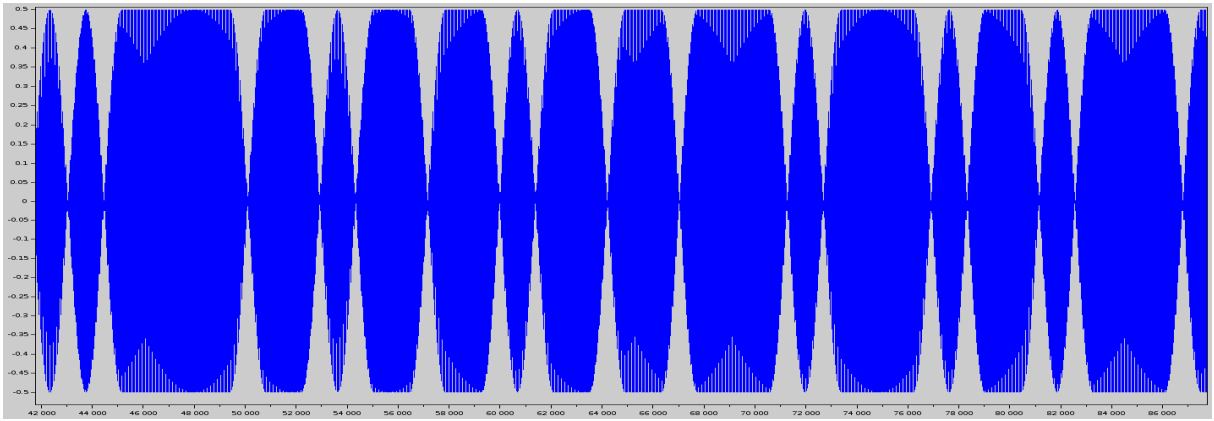


Fig. 3.40. BPSK31 test vector

To start the demodulation, the decoder should synchronize with the signal in order to so that the surrounding signal is enough. If we take a look to the signal in detail, we will realize that there are signals of two different frequencies:

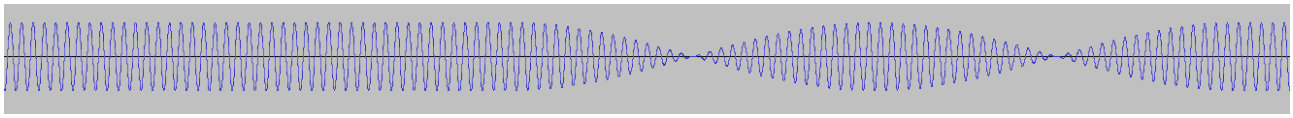


Fig. 3.41. BPSK31 test vector zoomed

One of them is the carrier which have a higher frequency and the other one is the signal that gives the BPSK31 baud signal, which variates at a frequency equal to baud rate which is 31,25 Hz. For this reason, we filter the first signal to left the one that we are interested on, to do that a low pass filter was created the desired signal has a frequency proximal to 31,25Hz but we have to set the cut frequency some Hz upper to avoid the stop band to attenuate the desired signal. The chosen frequency was 500Hz because it can be useful for most of the digital modulations.

To implement the 500 Hz filter knowing that the samples inside our system are usually sampled at 44100Hz. For this reason, in order to get a signal that is changing at 500 Hz, the signal should variate 44100Hz/500Hz = 88,2 times slower in order to make the filter faster and easier. We will say that is just 88 times slower the desired signal that the actual one.

To simplify the things, the positive and negative values are not giving any extra information that just positive or just negative, so we will just get the positive side taking the absolute value of each sample.

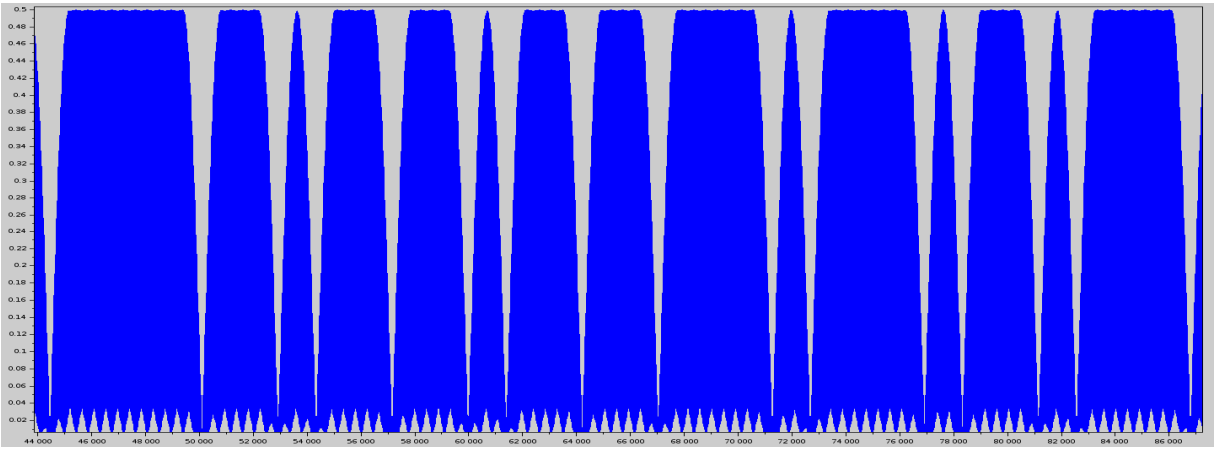


Fig. 3.42. BPSK31 test vector in abs value

Right now, we have more or less the desired signal but we need to filter it, to do that we just implement a LPF taking the 88 following samples and calculating the mean value. Doing that we obtain the following signal:

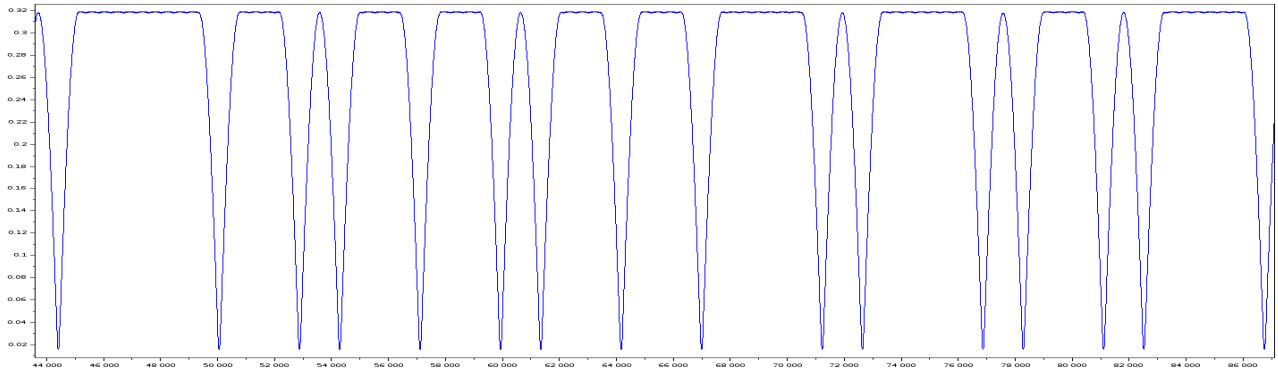


Fig. 3.43. BPSK31 test vector in abs value and filtered

We can see that the signal have some ripple but for our application is not relevant being just the 0,56% of the signal amplitude.

What is needed now is a starting moment to start sampling at sample rate in order to do that the signal should be synchronized using the first 0 found on the signal then the system start sampling at sampling frequency starting from the

sample that we get from sync and in the last step translate the 0's and 1's obtained to the corresponding character from the varicode table.

Once the demodulation code is achieved, the next step is to know is if this solution can accomplish all the requirements that we defined before for our system. The most important restriction is the real time processing, in order to do that, some tests where done (results are in the appendix) but mainly the important parameter that should be checked is the processing time knowing that the server is sending the audio at a rate of 44100samples /second in packets of 2048 samples.

$$\frac{2048\ samples/\ packet}{44100\ saples/s} = 0{,}046439909\ s/\ packet \qquad \textbf{(3.11)}$$

It means that, this is the maximum time that we have to take in order to process the sample before the next one arrives what will provide a "real" time system.

The time to filter the signal is the most restrictive one are 0,127 s the time to synchronize the starting sample is 0,002s and to get the bit stream the system takes 0,001s. This times are taken from the tests done in the Scilab test code, what means that is probably going to change depending on the language is chosen in the final implementation. The transition from the bit stream to the character output couldn't be measured due some strange errors in the code.

From this first results, we can say that the system doesn't accomplish with the requirement of process the sample in less than 0,046439909s, to improve the time of processing we can just drop some of the samples meanwhile the Nyquist theorem ($f_s \geq 2 \cdot f_{max}$) still being accomplished.

We can do an approach suffices to know how many samples we can reject to fulfill the real time condition.

$$\frac{0{,}13}{0{,}046439909} = 2{,}799316406 \sim 3 \qquad \textbf{(3.12)}$$

Knowing that, it will take some more time because is a last part pending to test (the translation from bits to characters). We will take ¼ of the samples in order to improve the processing time.

Doing that, the system will improve the filtering time to 0.03 meanwhile the synchronization time and the decoding time is not affected. But in the moment with this down sampling the system can achieve the "real" time requirement.

CHAPTER 4. CONCLUSIONS AND FUTURE LINES OF WORK

The future passes through the software defined radio and that is one of the main reasons why is so important to give to EETAC a WebSDR. This project have been deployed to accomplish all the objectives but this is just a base for the people that come to make their own versions and modify the system in order to improve it or just modify for some specific functionalities.

As is said before, this is a base so we can find hundreds of future lines of work we will explain just a few:

Integrate the BPSK31 modulations to the system and/or demodulate other digital signals

This is an interesting line of work because this system can be the first webSDR that decodes digital modulations.

Test in portable device

This is a free software based project what will allow everybody to have access to the source code and modify it without problems. For that reason, we can modify somehow the code or even use it in a small device like a raspberry what can allow us to create a low cost spectrum analyzer or a portable receiver.

Change the dongles

Always, is a good option to test the different receivers in order to compare. We have tested two different ham dongles, but is possible to try with some USRP or some other more professional receiver board.

Design a dongle

There are some hams that are starting to develop his own dongles and this in some years will be a normal thing and is easy to find the principal blocks for each part of a dongle needs but a high electronics knowledge is required to do that.

Tune the receptors on interesting bands for curiosity

The tuner can be set to some free used band in order to take interesting statistics or even to try to decode some type of not typical signals as the car remote or GSM, 3G packets.

REFERENCES

[1] Youngblood, G. A Software-Defined Radio for the Masses, Part 1(2002). Retrieved from: https://www.arrl.org/files/file/Technology/tis/info/pdf/020708qex013.pdf

[2] Software-defined radio. (n.d). In *Wikipedia*. Retrieved from: https://en.wikipedia.org/wiki/Software-defined_radio

[3] Software-defined radio. In The national association for amateur radio. Retrieved from: http://www.arrl.org/software-defined-radio

[4] Web Site: SDR radio. (n.d). Retrieved from: http://sdr-radio.com/

[5] Cass, S. A $40 Software-Defined Radio: A repurposed TV tuner can reveal a wide swath of spectrum (2013, June,25). Retrieved from: http://spectrum.ieee.org/geek-life/hands-on/a-40-softwaredefined-radio

[6] Blaine, J. (2009, July). An Introduction to Software Defined Radio. Retrieved from: http://www.ac0c.com/attachments/An_Introduction_to_Software_Defined_Radios_v2a.pdf

[7] Steven W. Smith. (2002) Digital Signal Processing: A Practical Guide for Engineers and Scientists. ISBN 0-7506-7444-X

[8] Web Site: FIR Filter Design. (n.d). Retrieved from: http://www.dspguru.com/dsp/faqs/fir/design

[9] WB5RVZ Software Defined Radio Homepage (2010, June 13). Retrieved from: http://www.wb5rvz.com/sdr/

[10]Project description (n.d.). In High Performance Software Defined Radio. Retrieved from: http://openhpsdr.org/

[11] SDR forum (2008). Request for Information on the Regulation of Next Generation Radio Systems. Retrieved from: http://www.sdrforum.org/pages/documentLibrary/documents/SDRF-07-RFI-0025-V0_0_4_Regulation_RFI.doc

[12] Anjum O., Ahonen T., Garzia F., Nurmi J., Brunelli C., Berg H. State of the art baseband DSP platforms for Software Defined Radio: A survey. EURASIP Journal on Wireless Communications and Networking (2011, June 6)2011:5 DOI: 10.1186/1687-1499-2011-5. Retrieved from: http://jwcn.eurasipjournals.springeropen.com/articles/10.1186/1687-1499-2011-5

[13] Web Site:TS-990S. (n.d.). Retrieved from: http://www.kenwood.com/usa/com/amateur/ts-990s/

[14] UbuntuHamsPackages. (n.d). In UbuntuWiki. Retrieved from: https://wiki.ubuntu.com/UbuntuHamsPackages

[15] Manninem T. gMFSK: A Gnome Multimode HF Terminal. Retrieved from: http://gmfsk.connect.fi/

[16] Official AX25. FTP server:ftp://ftp.hes.iki.fi/pub/ham/linux/ax25/

[17] Web Site: SPLAT. (n.d.). Retrieved from: http://www.qsl.net/kd2bd/splat.html

[18]Web Site: glfer. (n.d.). Retrieved from: http://www.qsl.net/in3otd/glfer.html

[19] Continuous wave. (n.d). In Wikipedia. Retrieved from: https://en.wikipedia.org/wiki/Continuous_wave

[20] James A. Cadzow, "Spectral Estimation: An Overdetermined Rational Model Equation Approach," *Proc. IEEE*, vol.70, no. 9, Sep. 1982.

[21] Jones, G. Packet Radio: Introduction to Packet Radio. Retrieved from: https://www.tapr.org/pr_intro.html

[22] Fldigi. (n.d.). In Wikipedia. Retrieved from: https://en.wikipedia.org/wiki/Fldigi

[23] W1HKJ & Associates (2017, January 20). Software. Retrieved from: http://www.w1hkj.com/

[24] Web Site: Ham radio deluxe. (n.d.). Retrieved from: http://ham-radio-deluxe.com/

[25] Wine (software). (n.d.). In Wikipedia. Retrieved from:https://en.wikipedia.org/wiki/Wine_(software)

[26] Wide-band WebSDR. Faculty for electrical Engineering, Mathematics and Computer science. University of Twente. Enschede-The Netherlands. Retrieved from: http://websdr.ewi.utwente.nl:8901/

[27] Web Site: ShinySDR (n.d.). Retrieved from: https://github.com/kpreid/shinysdr

[28] Ettus Research. a National Instruments Company. USRP™ B200mini Series. Retrieved from: https://www.ettus.com/content/files/USRP_B200mini_Data_Sheet.pdf

[29] Ettus Research, a National Instruments Company. USRP™ N200/N210 NETWORKED SERIES. Retrieved from: https://www.ettus.com/content/files/07495_Ettus_N200-10_DS_Flyer_HR_1.pdf

[30] Web Site bladeRF x40. (n.d.). Retrieved from: https://www.nuand.com/blog/product/bladerf-x40/

[31] Web Site: HackRF One.(n.d.). Retrieved from: https://github.com/mossmann/hackrf/wiki/HackRF-One

[32] Web Site: Airspy .(n.d.). Retrieved from: http://airspy.com/

[33] Web Site: Duncube Dongle Official Web Page. Retrieved from: http://www.funcubedongle.com/

[34] SDR (Software Defined Radio). In WikiStart. Retrieved from: http://osmocom.org/projects/sdr/wiki/Rtl-sdr

[35] WWW RTL2832U. (n.d.). Retrieved from: http://www.realtek.com.tw/products/productsView.aspx?Langid=1&PFid=35&Level=4&Conn=3&ProdID=257

[36] WWW Differences Between PCM/ADPCM Wave Files Explained. (n.d.). Retrieved from: https://support.microsoft.com/en-us/kb/89879

[37] Web Site: Official Open Web RX Official Web Page. Retrieved from: http://sdr.hu/openwebrx

[38] E.E. Azzouz, A.K.Nandi. Procedure for automatic recognition of analogue and digital modulations, IEE Proc-Commun, Vol. 143 No 5 October 1996

[40] R. Brault, R.Piat . Las antenas. ISBN: 84-283-1835-2 pages:86-87,140-141

[41] Frenzel, L. (2013, Jul 16) What's The Difference Between A Dipole And A Ground Plane Antenna? Retrieved from: http://electronicdesign.com/wireless/what-s-difference-between-dipole-and-ground-plane-antenna

[42] Munizaga, M. (2014, May 1). ANTENAS - CE2PNO: "Las mejores antenas dipolos HF... calidad, ideal para salidas de campo, emergencias y espacios reducidos". Retrieved from: http://ce2pno.blogspot.com.es/2014/05/configuracion-g5rv-jr.html

[43] Beltran, M. (2014, Jan 14). ANTENA DOBLE BAZOOKA, DISEÑO CONSTRUCCION Y EFICIENCIA. Retrieved from: http://10sd156.blogspot.com.es/2014/01/antena-doble-bazooka-diseno.html

[44] Web Site: Funcube Nanosatellite Official web page. Retrieved from: https://funcube.org.uk/

[45] Web Site: Unión de Radioaficionados Españoles Oficial Webpage.Retrieved from: http://www.ure.es

[46] PPLATO, University of Reading. Introducing waves. Received from: http://www.met.reading.ac.uk/pplato2/h-flap/phys5_6.html

[47] Andras Retzler, 2014, Software Defined Radio Receiver Application with Web-based Interface. BSc Thesis, Budapest University of Technology and Economics, Department of Broadband Infocommunications and Electromagnetic Theory.

[48]Web Site: Official Osmocom repository for the SDR-RTL controller software. Retrieved from: git://git.osmocom.org/rtl-sdr.git

[49] Web Site: Qthid Funcube Dongle Controller. (n.d.). Retrieved from: https://sourceforge.net/projects/qthid/

[50] Martinez, P. PSK31: A new radio-teletype mode with a traditional philosophy. Retrieved from: http://det.bi.ehu.es/~jtpjatae/pdf/p31g3plx.pdf

[51] Web Site: PSK31 SETUP AND OPERATION. (n.d.). Retrieved from: https://bpsk31.com/operation/

[52] RTL-SDR (RTL2832U) and software defined radio news and projects. Retrieved from: http://www.rtl-sdr.com/tag/sdrsharp/

[53] Web Site: Global Tunners Official software page. Retrieved from: https://www.globaltuners.com

[54] Jim Abercrombie, N4JA. July 2005. Understanding Antennas For The Non-Technical Ham. Retrieved from: http://www.hamuniverse.com/n4jaantennabook.html

[55] Owen Duffy. 2007. Double Bazooka Antenna performance. Retrieved from: http://f5ad.free.fr/Liens_coupes_ANT/G/VK1OD-Etude-des-Bazookas.htm

[56] Web Site: IZ2XMK, Building a full size dipole antenna for 80m band, Retrieved from: https://qrz.com/db/IZ2XMK/?mlab=

ACRONYMS

SDR	Software Defined Radio
WebSDR	Web Software Defined Radio
RF	Radio Frequency
ADC	Analogic to Digital Converter
DAC	Digital to Analog Converter
DSP	Digital Signal Processor
DVB-T	Digital Video Broadcasting — Terrestrial
IF	Intermediate Frequency
TX	Transmission
RX	Reception
AGC	Automatic Gain Control
OS	Operative System
ENOB	Effective Number Of Bits
COFDM	Coded Orthogonal Frequency-Division Multiplexing
RTL	Register-Transfer Level
IC	Integrated Circuit
USB	Universal Serial Bus
DDC	Direct Digital Control
PC	Personal Computer
I+Q	In phase and Quadrature
SWR	Standing Wave Ratio
TCXO	Temperature Compensated Crystal Oscillator
LNA	Low Noise Amplifier
OIP3	Third-order intercept point
ISS	International Space Station
VSWR	Voltage Standing Wave Ratio
JS	Java Script programming language
ADPCM	Adaptive Differential Pulse Code Modulation
BPSK	Binary Phase-Shift Keying
QPSK	Quadrature Phase Shift Keying
Freq.	Frequency
Ham	Amateur Radio Operator

APPENDIX 1. Installation Guide for Rtl-Sdr Dvb-T Dongle Controller

In order to control the RTL-DVB-T dongle we will use a library that works in almost all of the DVB-T dongles, which allows the decoder to stop giving a video output giving the I/Q samples of the signals.

In order to install the software first of all we should clone the repository with the following command:

```
git clone git://git.osmocom.org/rtl-sdr.git
```

Once we have the code on our computer the next step is to build the software, this software have dependencies of libusb1.0-dev.

Is recommended to build this code with cmake, we can just execute the following commands:

```
cd rtl-sdr/
mkdir build
cd build
cmake ../
make
sudo make install
sudo ldconfig
```

A good practice is also to install the udev rules to be able to execute the program without the root permissions to do this we should use the above code changing the cmake line for:

```
cmake ../ -DINSTALL_UDEV_RULES=ON
```

At this point code has been build and installed, in order to use it we can run the following instruction:

rtl_sdr -s [samp_rate] -f [center_freq] -p [ppm] -g [rf_gain] –

APPENDIX 2. Installation Guide For Funcube Pro + Controller Software

First of all we need to plug the dongle and without doing anything else type in terminal:
$ arecord -l

We have to download the qthid software from: http://sourceforge.net/projects/qthid/files/4.1/

Just uncompressing the files in a known directory and then copy the funcube-dongle.rules to /etc/udev/rules.d

(OPTIONALLY)
In order to create a desktop icon creating a .desktop file and copying the following code:

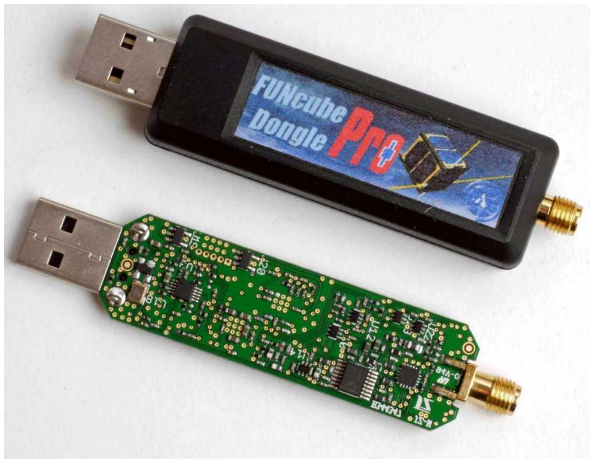
```
Name=qthid
GenericName=qthid 4.1
Comment=FCD Pro+ Controller
Exec=/home/youruser/folder_where_you_left_files/qthid-4.1-linux-i386/qthid
Icon=applications-debugging
Terminal=false
Type=Application
Categories=Network;HamRadio;
```

From terminal you should give execution permission to the file as follows:
sudo chmod +x qthid.desktop

ERRORS:
if you are executing a Ubuntu higher than 12 the library libudev.so.0 is not available any more in order to fix this we will use the library libudev.so.1 which is mainly the same, in order to make this change the easiest way is to generate a symlink, which is a simulated link we can do this by simply typing the following command on a terminal:

$ sudo ln -s /lib/x86_64-linux-gnu/libudev.so.1 /user/lib/x86_64-linux-gnu/libudev.so.0

This will call the libudev.so.1 library each time that a program tries to load the libudev.so.0 library.



APPENDIX 3. Installation Guide For Open Web RX

First of all we need to install the following packages:

- libcsdr
- ncat

In order to do that we can run the following commands on terminal:

```
$ git clone https://github.com/simonyiszk/csdr.git
$ cd csdr
$ make
$sudo make install
$ cd ..
$ sudo apt-get install ncat
```

After the packages are installed we can now clone the main OpenWebRX repository:

```
$ git clone https://github.com/simonyiszk/openwebrx.git
```

Now we have all the necessary code in order to start our receiver, to do that we have to enter to the code directory and modify the configuration file:

```
$ cd openwebrx
$ [your favouriute editor] config_webrx.py
```

Now we should configure all the receptor parameters* and then to run the server we must introduce the following code on terminal:

```
$ python openwebrx.py
```

It should work now, just remember that probably if you are using a SDR-RTL device will have a small error on the frequency showed at the screen and the real receiving frequency.

* For the RTL SDR, the configuration should be:

```
print(center_freq_corrected)
start_rtl_command="rtl_sdr -s {samp_rate} -f "+str(center_freq_corrected)+" -p {ppm} -g
{rf_gain} -".format(rf_gain=rf_gain, center_freq=center_freq, samp_rate=samp_rate,
ppm=ppm)
format_conversion="csdr convert_u8_f"
```

And for FUNCUBE dongle we should the activate QTHID and configure the config_webrx.py as follows:

```
samp_rate = 96000
start_rtl_command="arecord -f S16_LE -r {samp_rate} -c2 -".format(samp_rate=samp_rate)
format_conversion="csdr convert_s16_f | csdr gain_ff 30"
```

APPENDIX 4. EETAC webSDR User Manual

On this appendix will be explained how to operate with EETAC webSDR.

Let's check the basic sections:

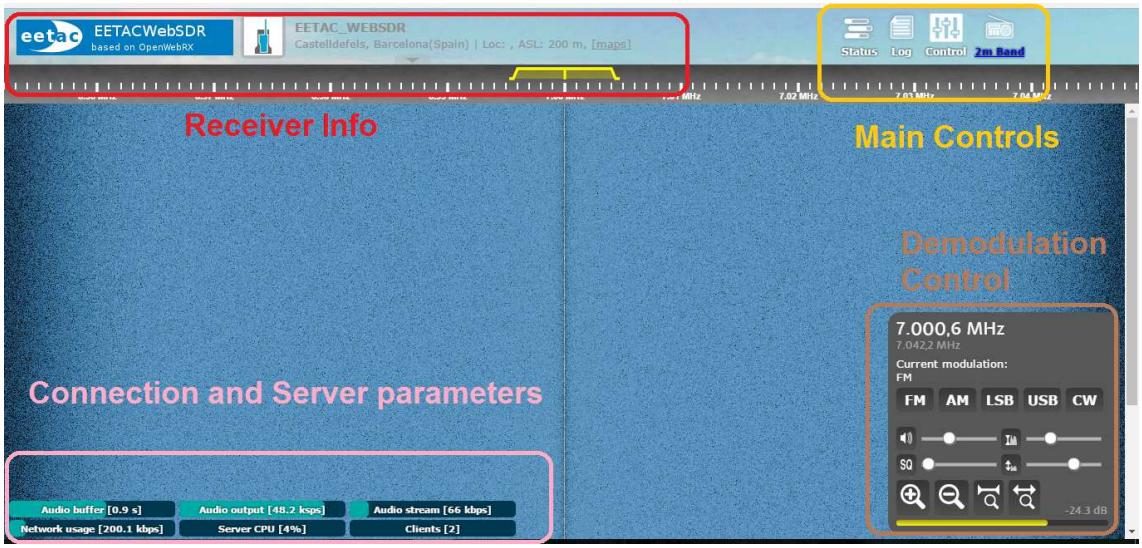


Fig 1 Main EETAC webSDR view

Also we have the frequency bar which is the gray bar that is under the receiver Info and the main controls, on this bar we can select a frequency or clicking the mouse on some point on the waterfall and then our filter will automatically be centered at that position. The filter is represented as the yellow bar inside the frequency bar is showed on detail below:

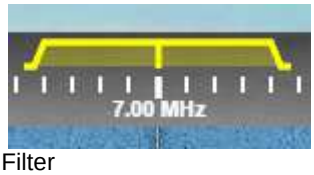


Fig 2 Filter

If we want to modify the filter bandwith we can just click and drag one of the lateral bars of the filter untill the desired frequency.

The receiver info bar shows information introduced manually by the radio operator such as receiver antenna or the receiver location.

Fig 3 Detail of the receiver info bar.

Fig 4 Main controls detail.

The main controls are basically to show or hide the submenus and other elements like the client log, which is a console that appears on the left side, but right now no interesting info is plotted there, so we can omit this element.

We can also show or hide the connection and server parameters, there some parameters are showed with a bar, parameters like the network usage of this connection an approximate server CPU usage, the number of clients, and some audio information.

Fig 5 connection and server parameters detail.

Also the main controls have a button to change between the 2m and the 40m band and a button for showing or hiding the demodulator control.

The demodulation control shows the central frequency of our filter (big white text), the text below (grey text) is the frequency position of our pointer, and the current modulation information.
Also we have the demodulation options and some receiver parameters, as the output volume, the squelch level, the maximum and minimum waterfall level and some zoom options, which can be also done with the mouse wheel.
On the bottom of this panel we can find the received signal level in [dB].

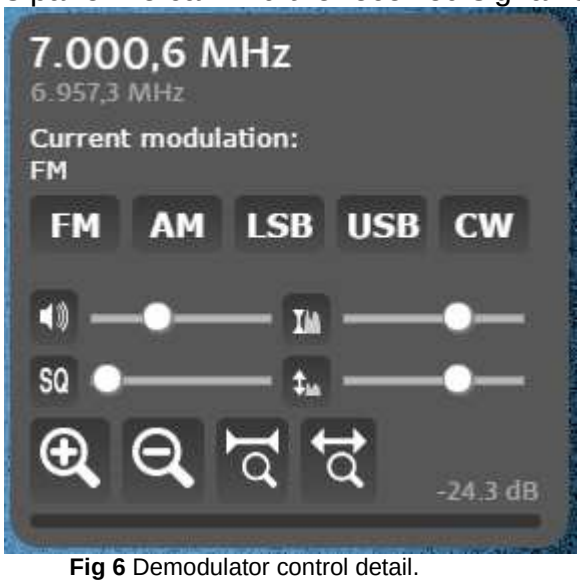
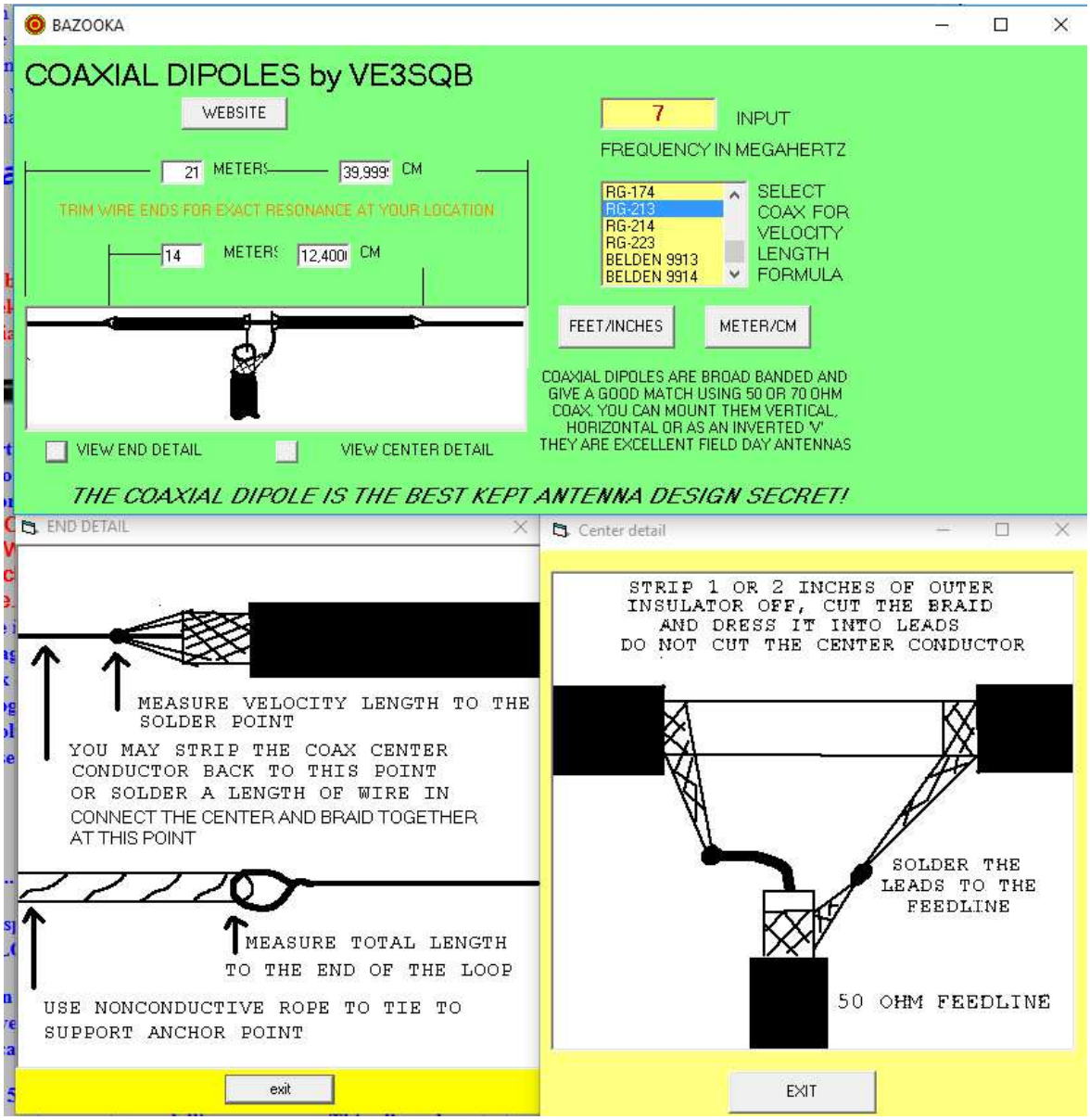


Fig 6 Demodulator control detail.

APPENDIX 5. Bazooka Simulator Antenna Size Orientation



APPENDIX 6. Server Tests For SDR-RTL

a. Stress Test

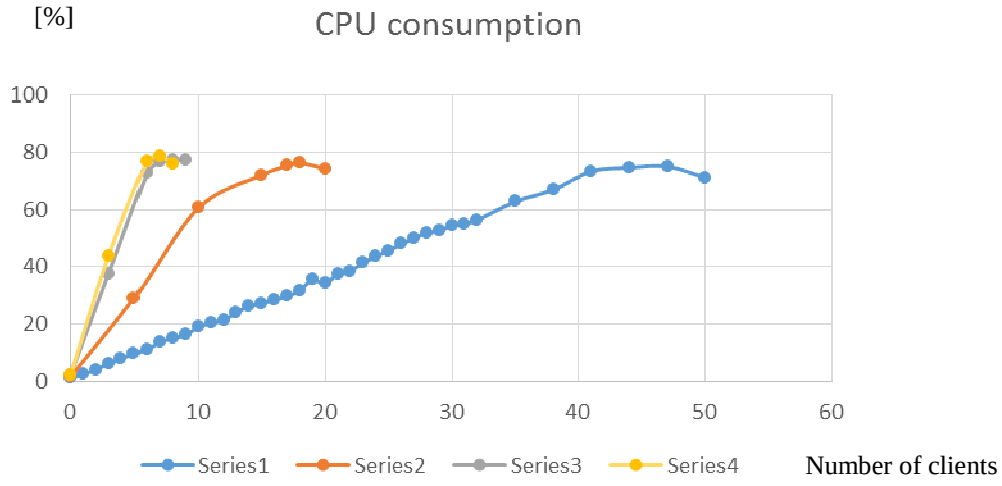
	250000	0,22MHz	
Clients	RAM	SWAP	CPU (%)
0	1005844	0	1
1	1013148	0	2,3
2	1019908	0	4
3	1069104	0	6
4	1075956	0	7,8
5	1084436	0	9,6
6	1090344	0	11,1
7	1097148	0	13,7
8	1103540	0	15,1
9	1111628	0	16,3
10	1118821	0	18,9
11	1125488	0	20,4
12	1131304	0	21,3
13	1140676	0	24,1
14	1144528	0	26,2
15	1150468	0	27,3
16	1157288	0	28,4
17	1165692	0	29,7
18	1171960	0	31,8
19	1175996	0	35,5
20	1182460	0	34,2
21	1188976	0	37,6
22	1194780	0	38,6
23	1202008	0	41,5
24	1209424	0	43,8
25	1215700	0	45,6
26	1221400	0	48,1
27	1228928	0	50
28	1238812	0	52
29	1244276	0	52,8
30	1258724	0	54,4
31	1252832	0	55,1
32	1259688	0	56,5
35	1283927	0	62,9
38	1304006	0	67,1
41	1323256	0	73,5
44	1344620	0	74,8
47	1377523	0	75,2
50	1429652	0	71,2

	1024000	1,22MHz	
Clients number	RAM	SWAP	CPU (%)
0	1064360	0	1,5
1	#N/A		#N/A
2	#N/A		#N/A
3	#N/A		#N/A
4	#N/A		#N/A
5	1096406	0	29,2
6	#N/A		#N/A
7	#N/A		#N/A
8	#N/A		#N/A
9	#N/A		#N/A
10	1134112	0	60,7
11	#N/A		#N/A
12	#N/A		#N/A
13	#N/A		#N/A
14	#N/A		#N/A
15	1260123	0	72,2
16	#N/A		#N/A
17	1266068	0	75,5
18	1256173	0	76,5
19	#N/A		#N/A
20	1286648	0	74,4

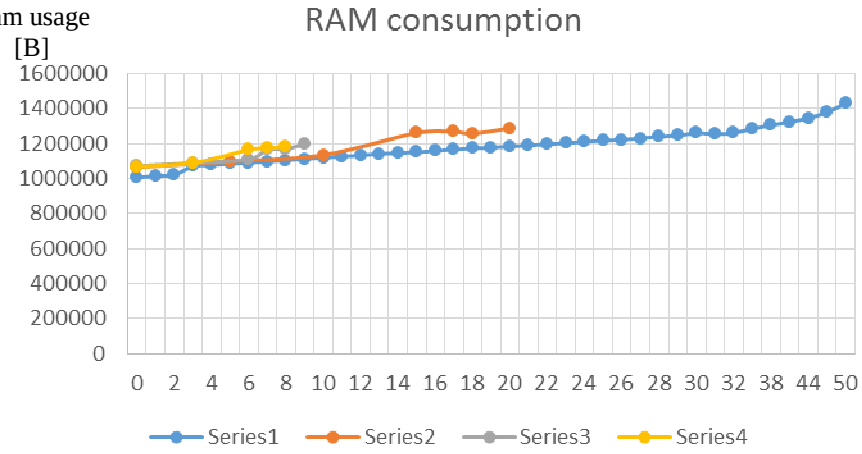
	2048000	2MHz	
Clients number	RAM	SWAP	CPU (%)
0	1070752	0	2,2
1	#N/A		#N/A
2	#N/A		#N/A
3	1088072	0	37,7
4	#N/A		#N/A
5	#N/A		#N/A
6	1108909	0	73
7	1164532	0	77
8	1168696	0	77,6
9	1198040	0	77,4

	2400000	2,4MHz	
Clients number	RAM	SWAP	CPU (%)
0	1067872	0	2
1	#N/A		#N/A
2	#N/A		#N/A
3	1088212		43,9
4	#N/A		#N/A
5	#N/A		#N/A
6	1164260	0	77,2
7	1174100	0	78,8
8	1179072	0	76,2

CPU usage [%]



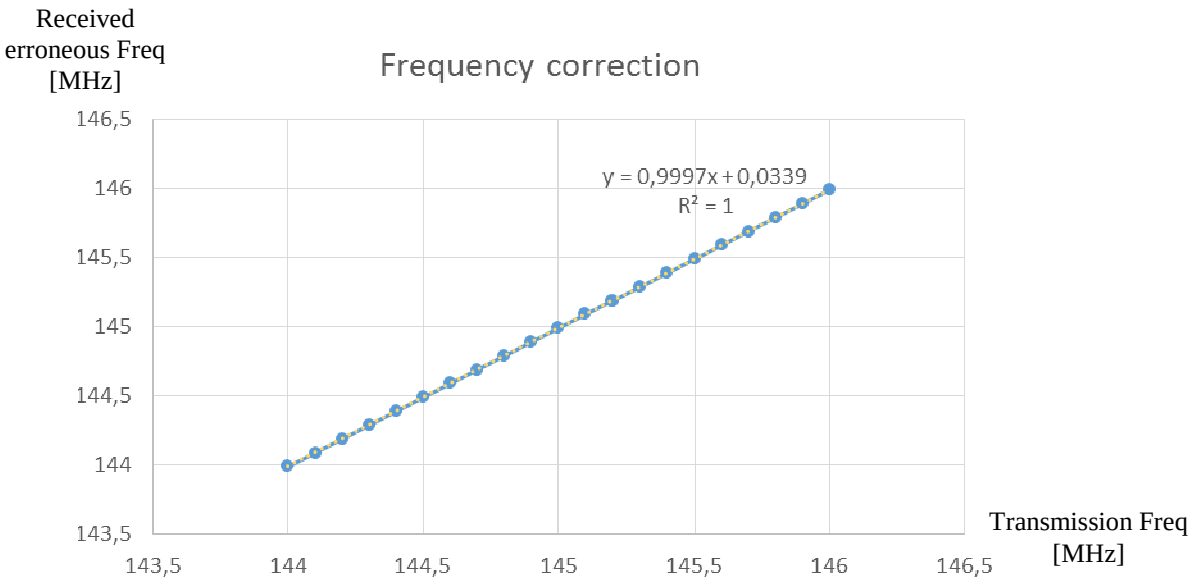
Ram usage [B]



We should remember that the server has a Intel Pentium D of 3GHz and 2 GB of RAM memory.

b. Frequency Deviation

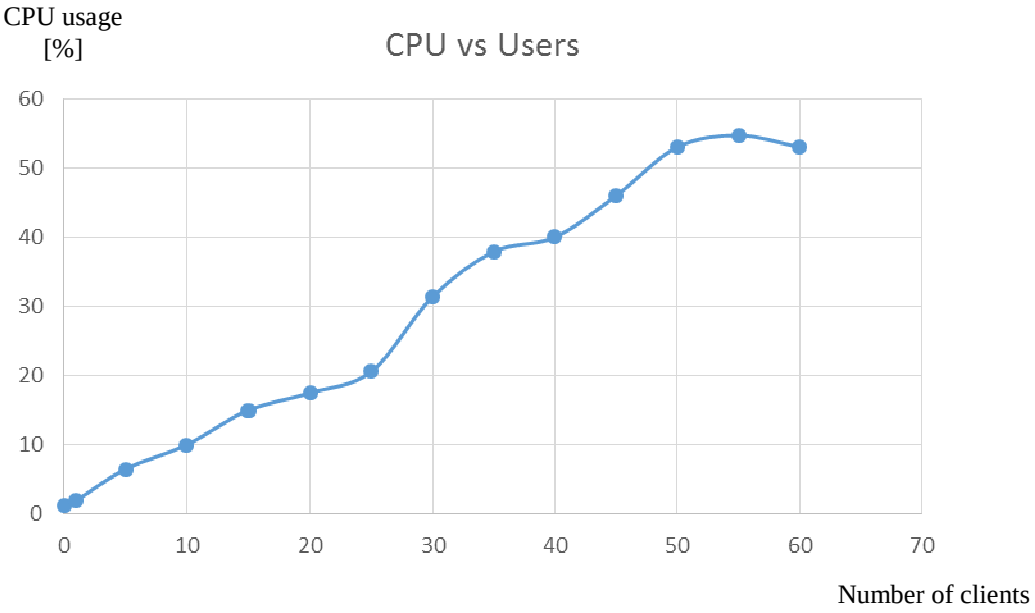
Transmitted Freq [MHz]	RTL Received Freq [MHz]
144	143,9882
144,1	144,0883
144,2	144,1883
144,3	144,2882
144,4	144,3882
144,5	144,4882
144,6	144,5882
144,7	144,6882
144,8	144,7879
144,9	144,8878
145	144,9879
145,1	145,0877
145,2	145,1879
145,3	145,2877
145,4	145,3879
145,5	145,4878
145,6	145,5878
145,7	145,6878
145,8	145,7878
145,9	145,8877
146	145,9877

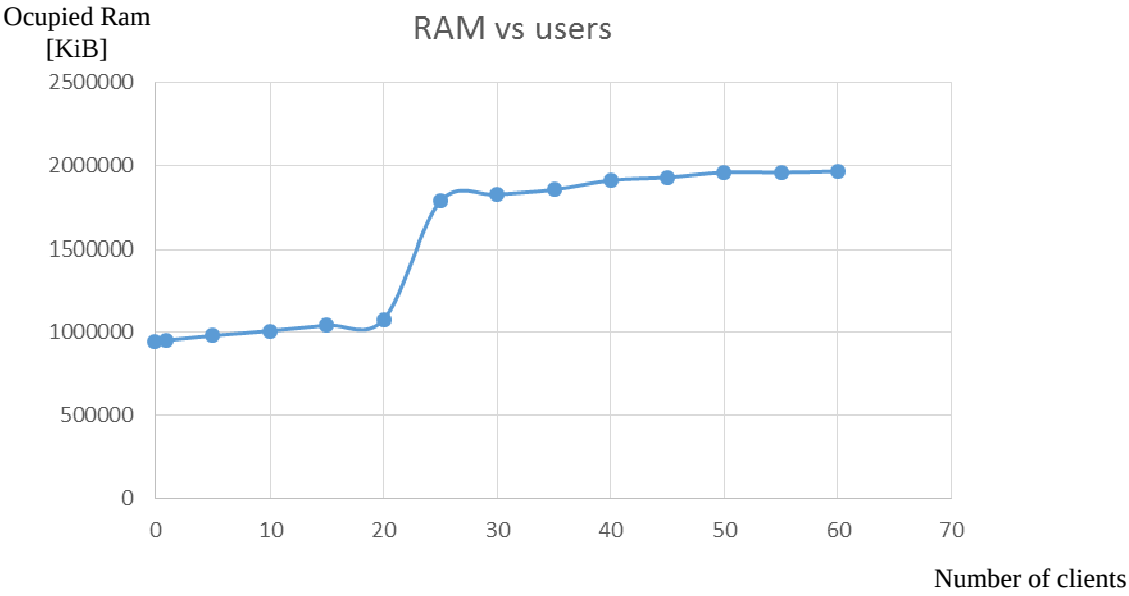


APPENDIX 7. Server Tests For FunCube Pro +

a. Stress Test

Number of active users	RAM	CPU
0	946772	1,3
1	953652	2
5	980980	6,5
10	1010444	10
15	1043972	15
20	1079004	17,5
25	1783368	20,6
30	1822480	31,4
35	1852244	37,9
40	1908609	40
45	1925805	46
50	1956596	53
55	1956092	54,7
60	1963175	53





As we can see on those graphics the progression of server usage is much less smooth than with the RTL-SDR dongle. We couldn't identify the reason of this.

APPENDIX 8. BPSK31 Scilab decoder

a. Scilab Functions

i. Filter

```
function [vector2]= digifilter(vector,samples)
    for i=1:(length(vector)-samples)
        media=0;
        for j=i:i+samples
         media = media + abs(vector(j));
        end;
        vector2(i)= media/samples;
    end;
endfunction
```

On this function a low pass filter is implemented, a samples is filtered using the following 88 samples, this 88 should be sent as a parameter to this function.

ii. Synchronization and Threshold decision

```
function [syn,th]= syncandthreshold(vector)
   lower = vector(1);
   greater = vector(1);
   pos =1 ;

   for i=1:(length(vector))
      if vector(i)<lower then
         lower = vector(i);
      elseif vector(i)>greater then
         greater = vector(i);
      end
   end;
   th = (greater+lower)/2;
   found =0;
   i=1;
   while found <1
      if vector(i)<th then
         lower = vector(i);
         for j=i:(i+352)
            if vector(j)< lower then
               pos =j;
               lower= vector(j);
            end;
      end;
      found = 1;
   end;
  i= i+1;
   end;
   syn = pos
```

```
endfunction
```

In this function, we process all the data in order to find the maximum and minimum value in order to determine an optimal threshold for the binary modulation, once the threshold is found, we find the first value that is below the threshold (some point of a 0 signal) and then we process a complete symbol (352 samples) data, searching for the lowest value, the position of this value will be the synchronization position.

iii. Obtain the Bits

```
function [vector2]= digitalyzer(vector,th,sync)
   i=1;
   j=0;
   while (i*352+sync+4*j)<length(vector)
      if vector(i*352+sync)<=th then
         vector2(i)= 0;
      else
         vector2(i)= 1;
      end;
      if modulo(i,5)==0 then
         j=j+1;
      end
      i=i+1;
   end;
endfunction
```

At this point we have to extract the data in order to obtain the bits, but we have a problem, the exact baud rate of the BPSK31 is 31,25 bauds/s what means that if we have a 11025Hz sound sample we have the center of a symbol transmission each 352,8 samples, but we can just take samples each 352 or 353 positions, in order to correct that we will take samples each 352 but after 5 samples we have accumulated a derivation of 4 samples so we can simply add them to correct this derivation.

b. Code Tests

This code has been tested in order to obtain some quality parameters, this are the results:

Noise=		10,00%		20,00%		30,00%		40,00%		50,00%		60,00%		70,00%		80,00%		90,00%		100,00%
(Noise in pecentage of signal peak power)																				
FILTER (time/packet= 0,127s)																				
BIT DECODING (time/packet=0,001s)																				
0	0	correct	0	correct	0	correct	0	correct	0	correct	0	correct	0	correct	0	correct	0	correct	0	correct
0	0	correct	0	correct	0	correct	0	correct	0	correct	0	correct	0	correct	0	correct	0	correct	0	correct
1	1	correct	1	correct	0	error	0	error	0	error	0	error	0	error	0	error	1	correct	1	correct
0	0	correct	0	correct	1	error	1	error	1	error	1	error	1	error	1	error	1	error	1	error
1	1	correct	1	correct	0	error	1	correct	1	correct	1	correct	1	correct	1	correct	1	correct	1	correct
0	0	correct	0	correct	1	error	1	error	1	error	1	error	1	error	1	error	1	error	1	error
1	1	correct	1	correct	0	error	0	error	0	error	0	error	0	error	0	error	1	correct	1	correct
0	0	correct	0	correct	1	error	1	error	1	error	1	error	1	error	1	error	0	correct	1	error
1	1	correct	1	correct	0	error	0	error	0	error	0	error	0	error	0	error	1	correct	1	correct
0	0	correct	0	correct	1	error	1	error	1	error	1	error	1	error	1	error	0	correct	0	correct
1	1	correct	1	correct	0	error	0	error	0	error	1	correct	1	correct	1	correct	1	correct	1	correct
0	0	correct	0	correct	1	error	1	error	1	error	1	error	1	error	1	error	0	correct	0	correct
1	1	correct	1	correct	0	error	0	error	1	correct	1	correct	1	correct	1	correct	1	correct	1	correct
0	0	correct	0	correct	1	error	1	error	1	error	1	error	1	error	1	error	1	error	1	error
1	1	correct	1	correct	0	error	0	error	0	error	0	error	0	error	0	error	1	correct	1	correct
0	0	correct	0	correct	1	error	1	error	1	error	1	error	1	error	1	error	0	correct	0	correct
1	1	correct	1	correct	0	error	0	error	0	error	1	correct	1	correct	1	correct	1	correct	1	correct
0	0	correct	0	correct	1	error	1	error	1	error	1	error	1	error	1	error	1	error	1	error
1	1	correct	1	correct	0	error	0	error	0	error	0	error	0	error	1	correct	0	error	0	error
0	0	correct	0	correct	1	error	1	error	1	error	1	error	1	error	1	error	1	error	1	error
1	1	correct	1	correct	0	error	0	error	0	error	0	error	1	correct	1	correct	1	correct	1	correct
SYNCHRONIZATION POS (time/packet=0,002s)																				
24		24		24		11		11		11		11		11		11		16		16
reference		correct		correct		error		error		error		error		error		error		error		error

As is shown on the table, the results are not bad, we have no error until the noise reach an amplitude of 30% of our signal so it means that in that case we will have a SNR of 11,111 what is approximately 10 dB.

Also the processing times has been taken into account, because is an important requirement to implement a real time program, we should try to optimize the functions to be as fast as it’s possible.